A Low Cost, Real Time, Robot Vision System,

with a Cluster-based Learning Capability.

A thesis presented for the degree of

Doctor of Philosophy

by

Nicholas Roger Schofield.

University of Durham

School of Engineering

and Applied Science

April 1988

ABSTRACT

This thesis describes the visual inspection techniques developed in a research project initiated by British Airways in conjunction with the School of Engineering and Applied Science at Durham University. The aim of the project has been to automate certain aspects of the catering operation at Heathrow airport. The first vision task is a purely inspection one in which a vision system is used to determine whether an in-flight meal tray has been correctly loaded by a robot. The second and main task, from a vision perspective, has been to develop a working approach to the identification and orientation of randomly arranged recyclable items on a tray returning from use on an airliner. The eventual objective of this task is to use this information to guide a robot to unload the tray automatically.

The brief for the vision system has been to produce a versatile low-cost industrial vision system, capable of performing the required inspection tasks, within a limited processing time, and requiring no specialized image processing hardware.

The first section of the thesis describes vision systems in general, the chosen system hardware, and the simple template matching based software developed for the first task.

The second and main section of the thesis details the development of software techniques to find a working approach to the second task which approximately falls into the robot vision catagory of a bin-picking operation. This software includes a new approach to industrial image processing,involving the use of cluster analysis techniques to provide an unsupervised object-learning and high speed recognition capability on standard microprocessor hardware. The approach is based around deriving object information from artificially generated shadows and includes high-speed edge and feature extraction algorithms and cluster analysis based techniques for object prototype determination and recognition. The section finishes with an assessment of the performance of the complete system and suggestions for consideration in further work.

ACKNOWLEDGEMENTS

I would like to express my gratitude for the invaluable technical support and assistance given to me throughout the project by the staff of the Durham University Microprocessor Centre, namely Peter Baxendale, Melos Kolar, Steven Teesdale and Jim Swift. I would also like to thank my highly literate departmental supervisor Dr. Clive Preece for his guidance and encouragement in the production in this thesis. Finally I would like to thank my many good friends for making my protracted stay in Durham enjoyable and doubly worthwhile. In this respect I am especially grateful to Dick, Sue and Carol.

PAGE NUMBERING AS ORIGINAL

CONTENTS

LIST OF FIGURES

Figures run consecutively following the bibliography ending page 147.

Appendix Figures.

STATEMENT OF COPYRIGHT

DECLARATION

The work contained in this thesis has not been submitted elsewhere for any other degree or qualification and that unless otherwise referenced it is the author's own work.

CHAPTER 1

AUTOMATED VISUAL INSPECTION

1.1 Introduction.

This thesis describes the visual inspection techniques developed in a research project initiated by British Airways in conjunction with the School of Engineering and Applied Science at Durham University. The objective of the whole project has been to develop a working approach to the automation of certain aspects of the British Airways catering operation at Heathrow Airport. The processes considered for automation are the loading of in-flight meal trays with food and utensils, and the stripping of the meal trays on their subsequent return from use on an airliner.

The brief for the vision system has been to produce a versatile, low cost industrial vision system capable of performing the required inspection tasks within a limited processing time, and requiring no specialized image processing hardware.

The thesis begins with a section which contains a definition of some visual inspection terms and discusses the potential value of automated visual inspection to industry in general. A range of particularly advantageous applications is given with a description of the limitations of currently available commercial vision systems.

The thesis is then divided into two main sections. The first smaller section describes the first vision project, – that of developing an inspection technique to determine whether a robot loaded tray has been correctly loaded. The approach taken to the problem is described. This is followed by a description of the vision system hardware, which is used throughout the research work, and the software developed for this first project. The section ends with a results and conclusions chapter which contains an assessment of the performance of the vision system as a whole.

The second section and main part of the thesis relates to the second project. This was the development of a working approach for the vision guidance of a robot to remove the recyclable items from a meal tray returning from use on an airliner. Essentially the same hardware was used in this second project as was used in the first project. For this reason this section of the thesis is mainly concerned with the

software developed for the system. This software includes a new approach to industrial image processing, involving the use of cluster analysis techniques to provide an unsupervised object-learning and high-speed image-processing capability. Without the use of specialized hardware the software based system capitalizes on the significant improvements that have been made in microprocessor performance and the decreasing cost of standard microprocessor hardware. These factors permit the real-time application of techniques from the fields of computer science and statistics. This section concludes with an evaluation of the system. The appendix section at the end of the thesis contains a more detailed software description in conjunction with program listings of the software developed. Following this there is a reference section which lists papers and books referenced in the thesis, a bibliography, and a section containing all the figures referred to in the text.

1.2 An Overview of Machine Vision Systems: Their Advantages and Limitations.

The inspection techniques developed in this thesis, for the British Airways project, are techniques suitable for a variety of automated vision applications. In this chapter the potential advantages offered to industry by such techniques are described in general terms. This is followed by an indication of the limitations of presently available commercial systems.

Before these descriptions there is a definition of some generally accepted terminology used here in this chapter and throughout the thesis.

1.2.1 A definition of terminology.

Automated visual inspection refers to the automatic inspection of a process, item, or range of items by some visual, non-contact means. Although the approaches and requirements of systems to perform the inspection may vary greatly they all involve as a first stage the conversion of an optical image into a series of electrical signals. The optical image provided can range from that created by a focused lazer light on a photodiode array to a standard vidicon close-circuit television camera image of a randomly, diffusely illuminated scene. The remaining stages in the computer vision system are the electronic hardware to convert the electrical signals created by the sensors into a form suitable for computer processing, and the computer hardware and software which take the image and perform the necessary

identification, orientation or gauging tasks upon it.

Automated visual inspection encompasses a range of terms. These include machine visual inspection, computer vision, machine vision, and robot vision. It is fair to say that machine vision and computer vision are equivalent terms. The way in which the results from the process are used is basic in deciding whether a system is thought of as an inspection system or as a robotic system [1]. In an inspection system a part is assessed for its conformance to some a-priori criterion. This applies to process control as well as discrete part inspection. In computer vision applied to robotics the part quality is assumed to be acceptable or the degree of non-conformance known. Here the priority is the identification of individual parts in the field of view and how they are orientated. Further requirements may be to find a particular feature, for example a gripping point or a path of safest approach.

1.2.2 The potential value of machine vision to industry.

The advantages machine vision systems offer to industry are listed below:-

1/ Direct cost and labour savings.

2/ Improved product quality and consistency.

3/ Increased speed of manufacture and throughput.

4/ Versatility of vision systems increases the number of tasks suitable for vision processing.

5/ Improved dimensional accuracy.

6/ The vision system information is provided in a convenient digital form.

The application considered for the vision system will dictate which of the above advantages are being capitalised upon, but any single advantage may make the implementation of a vision system worthwhile [2][3]. A description of these advantages is given in the following discussion.

Direct cost and labour savings can be made by replacing human operators and sophisticated, supervised and automatic, contact gauging systems, with automatic visual inspection systems. Inspection is of considerable importance in manufacturing plants, where it has been estimated that on average nearly 10% of people and 5% of machines are involved in such tasks [4][5]. In the case of replacing

human operators it has been found that the average totally automatic final inspection machine justifies its capital outlay in approximately 1.5 years on a two shift basis (US figures)[4].This includes direct inspection labour costs and savings made on reduced warranty repairs.

The second point is that automated inspection offers improvements both in quality and in consistency. Electro-optical inspection means that each part is checked in a consistent manner and the inspection of a set of particular measurements is 100% true. Neither of these is possible with subjective human inspectors given to the normal human weaknesses of people required to do monotonous repetitive tasks.

The rate at which certain vision systems can process parts offers the potential for a great increase in productivity – for example Diffracto Limited – Canada have installed a system for in-line inspection of engine valves with 12 cameras [4]. They inspect all the pertinent dimensions of the valves at a rate of 4200/hour also providing information on valve surface defects. Purchase of this system was justified over a conventional approach principally because of high part rate and virtual lack of maintenance compared with the complex setup procedure and maintenance required with contact gauges.

A further instance where vision techniques can give increases in productivity is when parts are identified visually before sorting, thus reducing the level of complexity of mechanical sorting devices and increasing throughput.

Machine vision techniques offer advantages where human operator inspection or contact sensing is not practical. This situation may arise in the following cases :-

1/ Where the object to be inspected is too large.

2/ The complexity of the object is such that contact sensing or human visual inspection becomes impractical, as illustrated by the increasing use of computer vision inspection techniques for P.C.B. inspection [6][7].

3/ The measurement to be made is an awkward interior measurement. Such an ability to 'see' in small restricted areas where either there is difficulty for a human to see clearly or it is impractical to build a mechanical contact gauging system to perform the task, can offer considerable advantages. An example

of this is in robot welding applications [8].

4/ Where the measurement to be made is in an environment hostile to the presence of human operators – for instance in the presence of toxic chemicals or radioactive elements, or in situations of extreme heat, noise, or mechanical vibration, for example in turbine blade inspection [8][9].

The dimensional resolution required of a machine vision system depends largely on the particular application. For instance in a robot guidance application a standard vidicon T.V. camera can usually provide sufficient resolution, whereas if the task involves replacing existing contact gauges for an accurate dimensional measurement or if the system is to be used for surface flaw detection, an optically and electronically magnified photodiode system can be used. One such system for the inspection of pump vanes has been installed by Diffracto which offers a resolution of 4 millionths of an inch (0.1 microns) with a dimensional range of +/-10000 [4].

A further advantage of machine vision systems is that the information provided is in a digital form that can be fed back directly to control an NC machine or used to guide equipment such as a robot. It has been suggested that in order to allow flexible manufacturing systems to realize their full potential it is essential that an effective machine vision system be developed. Present day FMS systems are limited in their ability to sense the position, orientation, shape and surface composition or other characteristics of workpieces. Such information could to a large extent be provided by a machine vision system. Examples of applications in this area have already included;–

1/ Robot welding – where the edge to be welded is tracked visually either before or during the welding operation [5].

2/ Robot paint spraying operations – where a vision system processes an image of a car on a production line, identifies the car from a known group and tells the robot down the line which paint program to call up [10].

3/ The SCAMP system which is used in the British Robotic Systems Ltd. computer-aided-manufacturing project to orientate workpieces on lathes [11].

4/ In the automotive industry vision guided robots are used to stack crankshafts [12], to insert glass windscreens [13], and in general assembly roles [14][15].

1.2.3 Limitations of present commercial vision systems.

Having detailed the positive aspects favouring the introduction of machine vision systems here is a list of limitations of present day vision systems which are summarized in a report from the Tran Tech Corporation in America [16].

1/ They offer limited 3-D interpretation.

2/ They offer limited interpretation of surfaces.

3/ There is often the need for a structured environment.

4/ A lengthy processing time is often involved.

5/ The high cost of vision systems.

6/ The systems lack flexibility and require skilled operator intervention.

Just as each particular application has its own particular requirements, so it may also involve any of the above limitations. A more detailed description of these limitations is given in the following section.

With regard to 3-D image interpretation as yet few stereoscopic multi-camera vision systems have progressed beyond the development stage [17]. Certain systems with a single camera are commercially available for 3-D gauging tasks which involve the use of motorized inspection beds [18]. It is also possible to derive 3-D information from a single camera by applying a transform function to the image based on reflected light intensity [19][20]. Again these programs have been developed in an academic environment and the resulting systems do not offer the robustness required for a system to operate in an industrial application. There has however been a large amount of research into application algorithms to work on pseudo 2.5-D images created by the application of structured light to 3-D images which has found successful industrial application [21][22][23].

In connection with the interpretation of surfaces there has been a great deal of effort expended in developing programming techniques in fields such as the automatic interpretion of aerial photographs [24][25][26], to reveal information about surfaces. Examples of these techniques include the deduction of texture and curva-

ture information from variations in reflected light intensity [19][27]. As yet little of this development has been applied to industrial applications, where there are obvious application opportunities for this technology. An example of this would be in surface flaw detection in polished surfaces by using textural analysis. It seems that large processing time overheads with current hardware and software have limited development for the industrial user. The need for a structured environment has been found to be a major limitation in machine vision systems. In order to get a clear image of the object, or objects, under inspection a specific lighting requirement is often imposed on the scene, for example strongly directional light, diffuse multi-directional light, or backlighting. The setting up of the inspection rig within a factory whose light is unstructured and variable can be an involved task. To a certain extent grey level processing can reduce the need for specific illumination conditions by using local intensity operators rather than global ones. This however, often incurs a significant processing time penalty. Large amounts of information and information manipulation are involved in image processing. When standard microprocessor hardware is used for this task processing time is often prohibitively long and precludes an industrial implementation. The required processing time savings, which may run into orders of magnitude, can be made in a number of ways including; more efficient algorithms, more efficient assembly language coding of such algorithms, and dedicated hardware for example facilitating parallel processing operations on the image.

Currently available commercial vision systems represent a large capital outlay when compared with a possible human alternative. A vision system operating on a suitably constrained image with appropriate lighting can be expected to cost between £8000 and £40000 depending on the sophistication of the system required [28]. Hidden costs can also be incurred as a result of the lack of hardware or software robustness. If for instance an error is made in a robot vision task this could be translated into errors in physical manipulations. Manipulators parts and fixtures could be damaged or destroyed. Other hidden costs can be in engineering the vision site for example for structured lighting conditions and in more elaborate fixturing of the part under inspection.

In the current climate of the micro-electronics revolution, rapidly decreasing costs and increases in the performance of individual integrated circuits will inevitably lead to dropping hardware costs and savings in processing time. From both the

software and hardware standpoints, as more companies enter such a potentially lucrative market, competition amongst them will lead to cheaper and more versatile vision systems [29][2][3].

The existing 'second generation' vision systems in industry lack flexibility in the sense that often skilled operators are required to set up and maintain a system which has no inherent learning capability. Situations like this may arise for example when the set of objects to be recognised is changed or expanded. A main focus of attention in the second part of the thesis is software development to remove this limitation.

In the majority of applications, vision systems will replace human operators. At present human vision out-performs machine vision in applications where work-pieces are examined at random, where a qualitative inspection is required, or where a small number of objects or scenes is to be interpreted. The most suitable applications for machine vision systems are in high volume manufacturing where a variety of parts is produced, each with readily distinguishable features and in which parts are presented for examination with a degree of 'structure'. This means they are not randomly placed but are presented, for example, in some sort of jig or fixture.

Having discussed vision system capabilities and limitations in general, in the following chapter we consider which of these factors are relevant to the particular applications.

CHAPTER 2

THE FIRST PROJECT

PROJECT OBJECTIVES AND METHOD OF APPROACH

In this chapter the overall objectives of the first project are described. The factors relevant to the implementation of vision systems are discussed in the context of the previous chapter. The approach adopted to overcome the problems presented by the first project is detailed. This is followed by a general discussion of the alternative vision system approaches that were considered.

2.1 The Project Objectives.

The vision work in this thesis can be considered as two distinct tasks, each with its own requirements. The first part of the thesis is concerned with the first task and also serves to introduce the hardware used throughout the research work. The second and main part of the thesis describes the more involved second task and is detailed in the second half of the thesis. A description of the first task follows.

In the first stage of the overall British Airways project it is intended that a series of robot cells will replace human operators in the preparation of meal trays for use on airliners. The prepacked items will be presented to a robot by a range of mechanical handling devices which then loads them into the trays. The vision system will be used to provide automatic quality control to ensure that the trays have been correctly packed in what can be considered a wholly inspection task.

The developed project sequence is as follows; the robot loads an empty tray with a cup containing a milk carton. The robot then loads a plastic bowl with a transparent plastic cover containing a variety of food, a sealed polythene bag containing cutlery and salt sachets etc. and a plastic sideplate containing a bread bun and a butter pack. After being loaded with a further open bowl the tray is released to travel down a roller conveyor which transfers it into the inspection station. An inspection is then performed and an indication made as to whether the tray has been correctly loaded. If it has not been correctly loaded the vision system

must give some indication of the particular loading error.

2.2 Justification of the System.

In the context of the previous chapter the justification for a vision system in this first application revolves around the need for high speed, non-contact, inspection capability. The only alternative to this approach is the use of human operators and hence the assessment of the viability of such a system must be made in comparison with this alternative. This means that many of the advantages offered by machine vision systems in general, for instance the ability to do high resolution gauging tasks and ability to function in awkward environments, are not relevant and the viability of the system must be assessed in terms of product throughput, consistency, and cost.

The rate of throughput in the first task is limited by the speed at which the robot can load the trays when working on a single vision system per cell basis, although the possibilty of linking a number of robots cells to a single inspection station was considered. In the single cell-per-inspection station case the processing time objective was 5 seconds for a complete inspection. This is in accordance with the minimum anticipated loading time of a tray by a single robot and compatible with that of the current human operators.

For the first task, in view of the simplicity of the task involved, it is not unreasonable to expect a high degree of consistency of inspection with a failure rate considerably lower than that anticipated for a human inspector performing the same task over long periods of time. The final, and probably most important criterion in these applications is the economic viability of the system. This must be assessed on the cost of the research and development undertaken and the cost of factory implementation against a significant reduction in labour costs. At present increasing labour costs and decreasing system costs indicate are that the application is becoming increasingly viable.

2.3 General Description of the Project System.

The hardware system chosen was a single camera, binary imaging system using strongly directional structured light. The light source was a standard visible spectrum spotlamp which was used in conjunction with a standard black and white vidicon close-circuit television camera. The camera was coupled to an Apple 8-bit microcomputer system which was in turn coupled to a dedicated Motorola M68020 microprocessor based system built 'in house' at the Durham University Microprocessor Centre. A detailed description of the actual hardware used is contained in Chapter 3.

2.4 Discussion of the Choice of Approach.

The following factors were considered when deciding upon the particular hardware and software for the first project.

1/ The system must be capable of verifying items, such as a bowl, containing a changing range of contents, standing on a tray surface. This task is complicated by the fact that the items, and in the worst case, their contents, are the same colour as the background tray. In addition they are of different surface finish and hence surface reflectivity to the tray.

2/ The inspection operation must be accomplished within a comparitively small time to make the application feasible.

3/ The objectives of the project must be achieved within a limited research budget.

It was clear that a colour vision system in all but the worst case – of empty items on a background tray of the same colour, would offer extra relevant information. When the additional cost involved with colour systems was considered however, the potential advantages were outweighed by economic considerations and a black and white approach favoured. This limited choice to vision systems dependant on reflected or transmitted light intensity. Practically this 'light' could be either ultra violet, infra red, laser, white, or coloured visible light. Laser systems have their usual applications in high resolution gauging, ranging and welding

operations, and involve considerable capital outlay. For these reasons they were considered unsuitable for the application. Although less prone to noise in the intended factory environment, the fact that the inspection station could be easily installed in a light-tight box meant that the additional cost and inconvenience of ultra-violet and infra-red sources and cameras outweighed the advantages, and a less expensive standard visible light system was used. (For completeness there is a brief description of alternative image gathering system hardware at the end of this chapter).

The items on the tray that must be verified are of the same colour and material as the base tray. This means that there is little noticeable variation in reflected light intensity in diffuse lighting conditions. For this reason structured lighting was investigated. It was found to be necessary to use some method of increasing the contrast around items on the tray to make it possible to verify their presence. The two approaches considered were backlighting the tray, and applying strong directional sidelight to the tray.

In the case of backlighting, the tray would need to be supported at points along the edge, and a strong directional light applied up or down through the tray opposite a light sensor. Thus for a British Airways tray of approximately uniform thickness the image would appear darker in areas corresponding to the places where items were on the tray surface. This approach has the following disadvantages:-

1/ There would be complications in jigging the inspection station with the tray on roller conveyors involving either feeding the tray onto a tight table/translucent conveyor or using some improvised side support for the sides of the tray on the conveyor, for example specially shortened side rollers.

2/ The tray was made of 3mm. thick, opaque cream plastic which meant that to get a satisfactory contrast on the resulting image an intense light source would be required. This would increase problems of scattered light around the tray which would possibly necessitate some form of extra cooling within the light box.

3/ There would be little or no indication in the image of multiple stacking of items due to the much reduced intensity of the transmitted light and the use of a binary threshold.

4/ The approach has less appeal in computer vision terms due to

its lack of similarity to human vision processing which most general vision system research is targetted towards. This may be a disadvantage when trying to reference relevant research material.

The chosen alternative structured light approach was to shine a spotlamp across the surface of the tray from one side. This had the effect of producing artificial shadows of items standing above the surface. This approach has the following disadvantages:-

1/ It is possible that exceptionally darkly coloured items placed on the tray would have the same light levels as the artificial shadows.

2/ Shadows of items placed close together or stacked could become linked together to produce complex compound shape outlines from which it would be difficult to deduce the component shadows.

3/ Only a single edge or two edges of a shape are available for its identification or location instead of a complete object outline.

This approach does however have the followiing advantages:-

1/ It provides a clear contrast between items of the same colour and was largely independent of surface reflectivity.

2/ It offers definition between stacked items provided the shadows do not overlap.

In the case of the latter approach it was assumed that the shortcomings could be minimised by more sophisticated software and the system was adopted in preference to the backlighting approach.

A further consideration was the type of light system to be used. The two alternatives were a binary light level system or a grey level system.

The processing flexibility offered by a grey level system has attractions especially in the case of a scene that is not evenly illuminated or has a variety of colours and surface types, or one in which surface orientation or shape information is to be derived from intensity profiles. This means that when a scene is coded with typically 16 or 256 grey levels, the integrity of edges represented by local maxima of rates of change of intensity is preserved. The price of this extra information is in more sophisticated and costly hardware, the need for a greater memory capacity, and in

order to make full use of this facility, more sophisticated software which increases processing time. The binary alternative chosen was modestly priced and performed adequately in the simplified, high contrast lightbox environment. As it is possible to adjust the capture threshold under computer control in the chosen system it would be possible generate a pseudo grey scale system by taking a series of images at different thresholds and merging them together. This could however only have been done at a processing time penalty and the possibility was not investigated.

Although the chosen system had the capability to switch between four cameras the possibility of stereo imaging was not considered. This could have been done by using two overhead cameras in a known physical arrangement with respect to the tray and by matching up identical points in the two 2-D images specific area and point coordinates could be calculated [17]. Another two camera method is to have one camera overhead and another at right angles looking across the scene [23]. These combine to give a restricted 3-D image. If applied to the project task this side camera would not have been able to provide much extra information due to the low profile of the project tray.

This multicamera approach was not investigated further, on the grounds that for the given application it would have introduced unnecessary complications into the resulting system.

2.5 Alternative Image Gathering Systems.

For completeness there follows a brief description of alternative image gathering system hardwares, and the reasons why they were not preferred. These systems include:–

1/ Laser scanning systems.
2/ Linear photodiode array systems.
3/ Solid state camera based systems.

Laser scanning systems are found more commonly in high resolution gauging applications. Due to the very large bandwidth (up to 85 Mhz) and low level of noise offered by photomultiplier tubes used as light transducers the resolution possible through a mixture of optical and electronic magnification has reached 0.1 microns. This is far higher than is required in our inspection or robotics application. These factors in addition to the high cost of the complex mechanism required to rotate the

polygonal mirror in order to deflect the light beam to scan the scene, and the cost of the lasers themselves, meant that such laser systems were not thought appropriate for the project application.

Linear photodiode arrays offer a number of advantages in common with RAM cameras over the chosen vidicon camera. These advantages include mechanical robustness, a significant size reduction, a potentially higher resolution with a bandwidth up to 35 Mhz, and the information in a form more readily processible by digital computers. They are significantly less expensive than vidicon and RAM cameras but unlike RAM cameras they cannot capture a complete image at one time. For this reason it is necessary to either move structured light with respect to an object or move the object with respect to structured light in order to build up a complete image of the object [21][22]. This introduces further unnecessary complexity and expense into the system and as a consequence such a system was ruled out.

Solid-state camera systems based on optic RAM chips have many of the advantages listed above in common with photodiode arrays in addition to the advantages that standard lenses can be used and a complete image gathered at one time. Modern high resolution RAM cameras have light sensitive two-dimensional arrays with element spacings of less than 5 microns which offer image resolutions of better than 512×512 pixels. The costs of such cameras has dropped dramatically in the last three or four years especially when compared with earlier image sensing chips such as the CCD. They are still however two or three times more expensive than the vidicon alternative. For this reason they were discounted as a hardware option.

(For a more comprehensive assessment of the characteristics of alternative image gathering systems see reference [30]).

CHAPTER 3

VISION SYSTEM HARDWARE

The hardware used in the first and second vision projects is shown in figures 3.1a and 3.1b respectively. The working system can be considered to consist of four separate parts. These are :-

1/ The lightbox/conveyor arrangement.

2/ The camera/digitizer/Apple microcomputer combination.

3/ The dedicated Durham University M68020 based system.

4/ The Cambridge Microsystems Codata M68000 based development system.

The photographic plate preceding the figures at the end of the thesis shows the combined microcomputer vision workstation built up for the projects.

3.1 The Lightbox.

The lightbox consisted of a wooden box which was light-tight except for the narrow roller conveyor entrance and exit (figure 3.2). The box prevented external light sources and changing ambient light changing the tray illumination conditions. This was especially important with the system set to operate in binary mode with a fixed light intensity threshold. The interior of the box was painted black in order to reduce the amount of scattered light from the bulb off the walls, as this causes a degradation in the quality of item shadow edges.

The light source was a standard 60 Watt internal reflector bulb which was mounted and angled to give clear shadow definition on the tray and an even tray illumination. The bulb height was chosen to reduce the difference in shadow width across the tray between those items casting shadows at the closest end of the tray to the light source and those furthest away from the light source. The angle was kept low enough however, to produce shadows of a distinct width (figure 3.3). For the first project the light source arrangement in figures 3.2 and 3.3 was satisfactory. For the second vision project it was found that in order to use the same convenient light source the spotlamp bulb had to be placed at a height of 8 feet above the rig

to provide a sufficiently parallel beam of light.

The overhead mounting of the camera was necessary in order to minimize the proportion of dark vertical interior sides of items and to control the amount of shadow visible to the camera.

In the first project the trays to be inspected were made of comparatively rigid moulded cream plastic, roughly rectangular in shape, about 40.5cm.× 27.5cm. × 1.7cm. in size and 3mm. thick (figure 3.4). Their surfaces had a number of indentations which acted as locating wells for items placed on the trays in order to prevent the items becoming disarranged in transit. In the second project a flat tray of the same size and colour was used.

In the first project the tray rolled into the light box on the roller conveyor. It settled against a stop and triggered a microswitch which in turn triggered the digitizer to grab a frame of the image. The test rig was arranged so that if the tray was rejected it was allowed to continue on down the conveyor. Otherwise it was pushed sideways by a pneumatic piston into a meal trolley stationed at the correct height alongside. In the trolley there were 11 shelves of 3 trays. The trolley was mounted on a hydraulic lift which started loading at the top of its travel and indexed down on filling each shelf. This process continued until the trolley was completely full when it was removed and replaced by another empty one.

In the second project, just prior to entering the light box, the returning tray passed through an air-jet station. At this point high pressure air was blasted across the surface of the tray to remove light-weight debris, for example napkins and paper sachets. Once the inspection had been made in the light box the tray was released to travel down a shallowly inclined conveyor to a robot unloading station. It was assumed that by avoiding rapidly decelerating the tray the items on it would not be disturbed in transit between these two points.

A further modification made to the standard light box shown in figure 3.2 in the learning stage of the second project was the use of a stepper motor driven rotary table in the light box. This table comprised a flat tray with a large circular disc cut out of the centre. The disk cut-out was then attached to a stepper motor at its centre and remounted, free to rotate, in its original hole (figure 3.2a). The control/drive signals for the motor came from a stepper driver card. This in turn received its control signals from the pulsed TTL output from the parallel interface

adaptor on a purpose-built parallel interface card within the Apple microcomputer.

This table allowed an item to be rotationally indexed under computer control whilst at the inspection station in the item learning stage of the second project. The table was removed when the light box was to be used for tray inspection.

3.2 Camera/ Digitizer/ Apple Microcomputer Combination.

The camera/digitizer/Apple combination was the image grabbing 'front end' to the vision system. The camera was an inexpensive National Panasonic WV 1500/B externally synchronisable black and white vidicon closed-circuit T.V. camera. The various forms of distortion that such cameras suffer from, for example astigmatism, and geomeric distortion due to a non-linear scan, were not significant in the low level resolution project tasks. No problems have been experienced with the camera either due to its vertical mounting, or the high temperature of its working environment.

The camera was used in conjunction with a Fujinon H6 × 12.5D zoom lens. This lens allowed a tolerance in the physical camera/workpiece arrangement, so that within given bounds the image could be zoomed to fill the camera's entire field of view. The lens has a manual iris which was set empirically to a small opening of F11, so as to give a large depth of field, and then kept constant throughout the sampling.

The link between the camera and the Apple microcomputer was provided by a Computech video digitizer unit. This unit converts the analogue camera video output signal into a digital form in the microcomputer memory. Once triggered the digitizer places a digital one in graphics memory corresponding to areas of the scene brighter than the set threshold and a logical zero where less bright.

The digitizer electronics are mounted on a card which fits into one of the auxiliary input/output slots of the Apple 2E microcomputer. There are two connections from the card to the camera: a composite frame and line synchronisation signal output line, and a black and white video signal input line. The video signal is converted at a rate of 25 frames per second and stored in one of the high resolution graphics screen memory areas of the Apple. The digitizer card operates by digitizing in a binary mode. This means that at regular intervals of the camera line scan the video line voltage is compared with a reference threshold voltage generated on the

card. If the differential comparator output is high that is the signal voltage is above the reference, then a digital 'one' is stored in a shift register and if not a 'zero' is stored in the register. The contents of this register are then made available to the Apple data bus at the correct time with respect to the Apple video circuitry reads and writes to graphics memory. The reference threshold voltage on the card can be set directly under program control or indirectly using the Apple game paddle potentiometers. The digitizer can be triggered either under program control or by external hard wiring for example by a microswitch.

In order to establish the range of brightness that the digitizer intensity scale of 0-255 corresponded to, it was first necessary to set two calibration potentiometers on the digitizer card. The first lower level potentiometer was set so that the image was completely dark when the camera was pointed at the darkest anticipated object and the second one was set when the image was just light when the camera was pointed at the brightest anticipated image. Once calibrated it was not necessary to make any further physical adjustments to the card.

In operation the digitizer was configured to capture images in its 'white' mode. In this mode only the light above a single lower threshold is set to logical 'one' bits. In addition, application software, provided on an Apple $5\frac{1}{4}$ inch disc with the card, allowed automatic logical operations between digital pages. These operations included logical ANDing or logical EORing of graphics pages under keyboard control. The software also provided for disc and printer operations.

When the digitizer is being operated modes and thresholds can be set and trigger signals provided under software control within the Apple, by either a BASIC or assembly language program.

The Apple 2E microcomputer system comprised a standard Apple 6502 processor board with on-board RAM and ROM, a video display interface and monitor, an attached keyboard, a disc interface with dual disc drives, and a series of interface slots for interface cards. In one of these slots was the video digitizer, in another the disc interface, and in another a purpose-built parallel interface card, replacing an earlier RS232 serial interface card.

The disc facility with its small access time offered convenient storage and recall of programs. The Apple video display system supported 80 column text presentation which although not essential in the project allowed a clear presentation

of the high resolution image provided by camera and digitizer.

The parallel interface was constructed in order to allow the rapid transfer of the captured image to the main processing facility – a dedicated Durham University Microprocessor Centre M68020 based system. This system was necessary because it was clear from an early stage that the program size and processing speed required was considerably beyond the scope of the limited 8 bit, 2MHz, 128k Apple. Originally the transfer to this system was via an RS232 serial link. This link was also used in the transfer of experimental images onto the host development system. For real time applications the maximum bit rate (no parity one stop bit) was only 9600 baud which without a handshaking protocol would still take:–

$$8192 \times (8 + 1 + 1)/9600 \approx 8.5 \text{ seconds.}$$

– to effect an 8Kbyte image transfer. In practice with handshaking it was found to take much longer – in the region of 20 seconds. As the amount of time available for completing the whole of either task was estimated at 5 seconds a more rapid transfer method was required. The parallel transfer card contained a simply buffered parallel interface adaptor connected to the Apple data and address buses with the Apple providing the necessary address decoding.

This parallel transfer with hardware handshaking has a theoretical maximum speed that is limited only by the 6502 processor speed in performing the necessary operations to transfer data from the correct part of graphics memory to the PIA on the interface card. For a software description refer to section 4.4.2.2.

An approximation of this transfer time is:–

Address setting instructions.	– 4 instructions per access
Data transfer instructions.	– 2 instructions per byte
Test of transfer success.	– 3 instructions per byte

Therefore number of instructions per byte transferred = 9

Number of bytes	= 8192
Number of machine cycles per instruction	= 3
Machine clock frequency	2 MHz.
Time for one instruction	1.5 μseconds.
Total theoretical transfer time for one image	= 1.5×9×8192μseconds.
	≈ 110 mseconds.

This theoretical value illustrates the greatly reduced transfer time possible by using parallel transfer techniques. In practice the transfer time was found to be around 0.3 seconds. The discrepancy can be accounted for by the the computing time overhead of a Basic language call to the subroutine and by the transmitted data not being promptly accepted by the M68020 system. In this case the handshake line would prevent the transfer success flags being set in the Apple output PIA and hence the loop to test transfer success would be repeated until the transfer had been achieved.

3.3 The DUMC M68020 Unit.

The dedicated unit was designed and built 'in house' at the Durham University Microprocessor Centre. The unit is based around the Motorola M68020 microprocessor chip running at 12Mhz. It contains a directly addressible random-access-memory of 1 megabyte in addition to a dedicated operating system RAM of 32Kbytes, and an operating system/monitor read-only-memory of 64Kbytes containing the monitor code and library subroutines. There is also provision for a user program ROM of 64Kbytes. In addition the unit has a single parallel and four serial interface lines. The parallel line was used for high speed image data input from the Apple. One serial line was connected to the host Codata development system and was used to download programs. Another serial line was connected to a QUME terminal which emulates a standardTVI920 terminal. This permitted keyboard communication with the unit itself and a monitor software facility also permits the direct communication of the terminal with the development system through the unit. The other serial lines were connected to a DUET-16 microcomputer with a high resolution graphics screen for a graphic representation of the results and an Epson FX100 dot-matrix printer which gave dot-matrix plots of images and a hardcopy of processing results. The 68020 unit also has dual 720 kilobyte disk drives for data and program storage.

3.4 The Cambridge Microsystems Codata Development System.

This is a multiuser system based around the Motorola M68000 microprocessor. In addition to 750k of RAM there is an 80 Mbyte Winchester hard disc storage unit with cartridge tape backup. The system runs the Unix look-alike Xenix op-

erating system with standard Unix facilities. In addition to an M68000 assembler there are compilers available for the 'C', Fortran, and Pascal computing languages. This system allowed the writing, development and debugging of the image processing programs in the user-friendly Unix environment, before downloading them onto the DUMC M68020 unit. The facilities provided included 'C' debugging aids and printing facilities.

For smaller programs it was possible to run the whole program in simulation on the system, with routines acting on test images downloaded from the Apple. In the later stages of the program development, especially for the second project, the limited amount of RAM on the system precluded this and only smaller program sections could be run at one time.

CHAPTER 4

VISION SYSTEM SOFTWARE.

4.1 Choice of the 'C' Programming Language.

The choice of 'C' as the main programming language was based on the combination of the facilities it provides. These include a clear control structure, a rich range of operators, and economy of expression [31]. The advantages are briefly described in the following section.

1/ 'C' has a range of control flow constructions –statement grouping, decision making, looping with loop termination at the top or bottom of the loop. It also has the ability to select one of a set of cases which provide the programmer with all the facilities to produce well structured programs and allow a modular nature to be maintained. This is particularly important in larger programs, of the size involved here and where the level of control within the program can often change.

2/ 'C' has a rich range of operators, including bitwise operators, which act on the same objects that computers do, specifically characters, numbers (of varying byte sizes) and addresses (through the use of pointers). This allows the array processing algorithms necessary for pattern recognition to be economically expressed and efficiently implimented at the compilation stage. The extensive use of pointers in 'C' is of great assistance in the production of fast/efficient M68020 machine code. When making the many array accesses, necessary in image processing, the base addresses of arrays are stored in one of the seven address registers provided by the M68020 architecture. In the 'C' implementation offsets are economically applied to these to make array element accesses.

The efficiency of compilation of 'C' has meant that the need to write sections of the program in assembly language has been avoided to a large extent. The only place it was considered a worthwhile return on the effort expended was in the time consuming thinning/outline extraction sub-function where a significant processing time saving of 20% was made as a result of optimizing the code in a highly repetitive task.

4.2 Choice of Compiler.

At the time of programming there was no compiler available on the system to produce code to take advantage of the extended instruction set of the Motorola 68020 processor on the dedicated processor unit. For this reason the existing code-compatible Motorola M68000 compilers were considered. There were two suitable 'C' compilers available on the M68000 UNIX Cambridge Microcomputers Codata development system. One was the host system's compiler and the other was a compiler from Whitesmiths Ltd. The latter was chosen because there was no documentation available on the host system compiler and its source code. This would have made it difficult to modify the interface subroutine calls for use on the dedicated DUMC 68020 board.

Because of the conciseness of 'C' the resultant code showed only small differences when programs were compiled using each of the compilers. This was seen when the two compatible compilations were stopped at the 'dot S' stage to show the resolved assembly language equivalent.

4.3 Programming Technique in General.

To make the introduction of visions system worthwhile in both project applications the main criterion of the software was speed of execution. Other considerations such as program compactness were of comparatively minor importance. For this reason care was taken to speed up the program wherever possible. Unnecessarily time consuming algorithms were avoided and substituted by faster, but possibly less elegant algorithms.

The general axioms adopted to achieve this high speed of operation are as follows :-

1/ The programs do not use a variable in a repetitive role that is of larger size than is necessary, for example using an unsigned character for a positive number not exceeding eight bits, an unsigned short word for a positive number not exceeding 16 bits.

2/ Variables are made unsigned when they were known to remain positive. This avoids the compiler having to include unnecessary sign-extend operations in the resultant code.

3/ The most commonly accessed variables in a function are made 'register variables'. These variables remain resident in one of the 68020 registers throughout the life of a function. They thus reduce the overhead of an access to a variable on the stack to that of a much quicker register reference. Due to the Whitesmiths' compiler restriction on the maximum of register variables to three, with no distinction being made as to whether they were normal variables (maintained in the data registers) or pointer variables (maintained in the address registers), it was found necessary in some circumstances to assist the compiler in producing fast code. The Motorola M68020 has seven usable data registers and six usable address registers which are completely independant of each other making this restriction unreasonable especially in the case of smaller programs. This limitation was bypassed by stopping the compilation at the assembly language stage (dot S stage), getting a listing and modifying the register allocation by hand.

4/ Avoiding the use of multi-dimensional arrays except in the most seldomly accessed arrays. This avoids the high multiplication computing overhead of accessing second and higher dimensional indices.

4.4 Software Developed for Project 1.

As stated earlier the two distinct project objectives mean that the software developed must perform two different tasks. This section contains a description of the overall program strategy to achieve the first of these project objectives. There is a description of the software development program including the development tools used in producing the final working version of the program. This is followed by a brief description of a proposed working version for industrial implementation.

4.4.1 The difference from ideal approach.

The straight-forward nature of the first project task meant that it was decided to use a variant of a 'template matching' technique. This approach is common in industrial image processing and involves the superposing of two images. One of these is the image under test and the other a pre-arranged image of an object in its ideal condition and/or its ideal configuration. In the project such images correspond to the tray under test and an'ideally' loaded tray. This ideally loaded tray would normally have all the particular items loaded centrally within their given locating wells.

For this type of processing it is essential that the two images are accurately registered with respect to the camera so that when the two images are overlayed descrepancies in the image are due to differences in the loaded items and not differences in the tray's physical position. This registration was, for project development purposes, provided by retracting pneumatic endstops on the roller conveyor.

The comparison of two such tray images is done either by logical ANDing or logical EORing (exclusive ORing) of the images. This involves microprocessor operations between equivalent areas of microcomputer memory storing the visual image to produce a third resultant image (figure 4.1). The ANDing process produces an image which highlights those point picture elements (pixels) in the computer graphics memory that the two images have in common, whereas the EORing process produces an image in which the differences in the images are highlighted. The former process was used in providing a noise reduced image of the ideal tray and the latter in the comparison of this image with the image of the tray under test. In deciding the suitability of a method it was necessary to consider the way the tray loading conditions are anticipated to be in error when a loading error does occur. These are:-

1/ In the loading process the robot would attempt to pick up a tray item from an empty or absent cartridge.

2/ The particular item picked could be lost by the robot gripper in transit or dropped prematurely on the wrong part of the tray.

3/ Limitations in the robot's repeatability would mean that over a period of operation there would be a drift in the positional accuracy with which the robot both picked and placed items. This drift may lead to the robot placing items on the tray with a positional accuracy lower than an acceptable tolerance limit for the tray to be considered correctly loaded.

Because of the nature of the loading operation it was considered that the likelihood of multiple loadings or stackings in the same place, or items being loaded in the precise position intended for another, was very small. Hence it was not considered necessary to test for these conditions in the inspection process.

With the requirement that checking is only to be for the above three errors it was decided that the 'ideal' tray image should be that of a completely empty tray.

The only features appearing on such a tray were shadows caused by tray indentations used to prevent tray items becoming disarranged in transit. In the case of EORing such an 'ideal' with a test tray image anything that is loaded replacing original shadow areas with light areas or obscuring originally light areas with new shadows will become highlighted.

In order to find where on the tray this difference was occuring, it was necessary to perform a count of pixels within certain regions of the resultant EORed image. The count for each of these regions was then compared with that obtained for a correctly loaded tray. If there was a discrepancy outside a given tolerance range then an indication was given on the DUMC unit monitor screen that there was an error in loading the particular item intended for that area. The computing method by which this pre-processing and counting was effected is given in detail in the program description.

4.4.2 Development program description.

Figure 4.2 is a flow diagram of the development program. This sequence comprises the following main program parts:–

1/ Picture acquisition software.

2/ An Apple image setup phase (including some image preprocessing.)

3/ Apple test image sampling and preprocessing.

4/ Apple / Microprocessor centre unit transmit and receive software.

5/ D.U. Microprocessor Centre unit image processing software.

The following sections describe each of these parts.

4.4.2.1 Apple microcomputer software.

4.4.2.1.1 Image acquisition software.(appendix AP1.2.1.1)

This is the software which 'grabs' the digital image by triggering the video digitizer within the Apple. In the program development stage the software allowed this triggering to be initiated by the user by pressing a game paddle button. The captured image was then stored in a section of the Apple high resolution memory.

These routines were called in both the set-up phase of program operation and at each individual inspection.

Although proprietory software was available to drive the digitizer directly this was written for keyboard operation only. No listing of this assembly language program was available to indicate either entry points into the subroutines or where the software itself loaded. As it was considered that it would take too long to disassemble this machine code new image acquisition routines were written.

The main Apple coordinating program, including the setting up and reading of the digitizer command and status registers, did not have to be performed at high speed, and a standard interpreted Apple Basic program was sufficiently fast for the purpose.

The program first set up the digitizer to operate in binary mode. The program then entered a main loop polling both the game paddle and button. The game paddle value was read in the range 0-255 and the value written to the registers controlling the digitizer threshold level.

The game paddle button was used to simulate the triggering of a microswitch on a tray's arrival at the inspection station. In the application the microswitch attached to the conveyor would trigger a softswitch within the Apple. When the button was pressed the digitizer was triggered and a single image frame stored. The user was then asked if the image was to be transferred to the M68020 system.

4.4.2.1.2 Set-up software.(appendix AP1.2.1.1).

The set-up phase produced the ideal image described in the introduction to superimpose on the test image. The Basic print statements instructed the user to place the empty tray on the inspection station. Two images were captured by calling the sampling routine described earlier.

On capturing each image the user was given the choice of retaking the image if it was not satisfactory. At this stage the binary threshold was set, to give best contrast between bright background and shadow by adjusting the game paddle. Once set on the first image the same threshold was kept throughout the sampling sequence.

The two images taken were then ANDed together by calling an assembly language subroutine (AP1.2.1.2). This works by accessing memory elements corresponding to equivalent points in the two images, test and 'ideal', and performing the logical 'AND' operation on them. This was done so that points on the shadow boundaries which were around the binary threshold and as a result appeared 'on' on

some images and 'off' on others, were significantly reduced in the resultant image (results Chapter 5.1). This image was then stored as the 'ideal' image and control was then passed to the sampling stage of the program.

The set-up phase needed only to be run once per series of a given tray to be inspected. In the development phase this occured each time a series of results was to be taken – with the camera set-up and digitizer thresholds being investigated. In the application program both these variables would be fixed and the set-up phase run only once for a given tray image at the start of the inspection run. Then, provided the set-up or tray arrangement does not change, the resultant 'ideal' tray image could be retained in computer memory.

4.4.2.1.3 Sampling and preprocessing software.(appendix AP1.2.1.1)

In the development stage the software allowed the user the option of retaking the test image before submitting it for processing. If the sample was satisfactory the 'EOR with ideal' assembly language subroutine was called (AP1.2.1.2). This worked in a similar way to the 'ANDing' subroutine. The resultant image was then offered for transmission to the DUMC M68020 system.

4.4.2.2 Apple/DUMC Unit interface software.(appendix AP1.2.1.3)

The parallel transfer link between the two systems incorporates full hardware handshaking. This means that the data transfer is 100% reliable and that no data is lost or corrupted. This considerably reduces the need for a more complex software transfer protocol. First a start of transmission databyte was sent followed by the fixed 8Kbyte image data block serially in rows starting at the top left corner of the image. Because of the memory mapping of the screen image in the Apple's high resolution memory most of the Apple assembly language routine was concerned with setting up the correct addresses from which to access the screen image. Once this had been achieved the data to be transferred was placed in the parallel transfer card's PIA(parallel interface adapter) TDR(transmit data register) and the status word polled to check whether the data had been sent. If it had, the address for the next data word was set up and the process repeated until all the data had been transmitted.

The image receiving function *aquipw()* (appendix AP1.2.2.4), in the DUMC unit, was called by the main coordinating function. It made repeated calls to the library routine *getpar()* to provide it with a series of characters from the parallel

interface port. The function first tested each character to determine whether it was a particular start character. If it was, the subsequent 7680 characters were read in from the port and placed in a linear array.

4.4.2.3 DUMC Unit software.(appendix AP1.2.2)

The unit is set-up by down-loading the compiled 'C' language M68000 code from the Codata development system in Motorola S2 format. The M68020 registers are set using the unit monitor program and the program instructed to run at a specified start address by setting the program counter.

The development software consisted of a main coordinating function which first declared and initialized the external variables used to pass parameters between the functions. It then called a series of central image processing functions and some optional functions selectable by the user for development and set-up purposes. These functions were:–

1/ A user print option and parameter modifying function.

2/ A greeting presentation function.

3/ The Apple interface function.

4/ An image array expansion algorithm.

5/ An optional image printing algorithm.

6/ One of three selectable area counting algorithms.

7/ A decision making function.

Refer to figure 4.3 for a flow diagram of the program. The first function called was *amodithr()* (appendix AP1.2.2.2) which allowed the user to select image printing options and area counting options for later in the program. After an introductory banner has been output on the monitor screen by the function *greet()* (appendix AP1.2.2.3), the interface routine *aquip()* (appendix AP1.2.2.4) already described was called.

In order to simplify later programming the 8Kbyte image storage array was next expanded into a 54Kbyte image array (seven significant bits) by calling the function *bitmatpw()* (appendix AP1.2.2.5). After this a single character of eight bits represented what was formerly a single bit. This meant that what were formerly bits could now be accessed and operated on as characters and not more awkwardly

through bitwise operations. Both the increased processing time cost and the increased memory storage requirement in doing this were considered acceptable. The next function to be called was the optional image printing routine *pripwb()* (appendix AP1.4.1.3). If called this function took the image as stored in the expanded array and performed the necessary manipulations on the data to produce a series of image bytes suitable for the vertically orientated dot-matrix printer head. The bytes were then output to one of the DUMC unit's serial ports connected to an Epson FX100 dot-matrix printer via calls to the *hwrite()* library subroutine.

Depending on the option set in the modify function one of three area counting functions was called next. The functions were similar, differing in the amount of information displayed on the VDU and hence in processing speed as the output of characters to the screen is a comparatively slow process.

The function *ufcf()* (appendix AP1.2.2.6) allowed the user to input the number of search windows and their coordinates from the keyboard to replace the default values used by the other count functions *ufcoupw()* and *ufcw()*. These windows defined the position and size of areas to be examined by the counting routine. Because of the comparatively regular rectangular shape of the shadows a series of rectangular windows were used. The windows were defined by cartesian coordinates of the upper left and lower right window corners with the x coordinate increasing from left to right and the y coordinate increasing from top to bottom. The way the user chooses these areas is described later.

All the counting functions took the specified coordinates of the count windows and calculated the corresponding areas to be checked in the image array. The function then preceeded to count the 'on' pixels in these areas and the resulting counts were stored in a form accessible to the subsequent decision function.

In the decision function *decpw()* (appendix AP1.2.2.7) the window counts deduced above were compared to the respective thresholds set for each area. If the count was below the threshold in a particular area of the tray the count was 'more like' the bright background tray and hence the tray was deemed to be incorrectly loaded. The tray was then rejected and a message displayed on the VDU screen to indicate in which particular aspect the tray had failed.

4.4.2.4 A typical operating sequence.

A typical development program sequence to inspect a tray is as follows :-

The Apple is set up by running the program and following the prompts appearing on the monitor screen. The program first produces the required ideal empty tray image by the logical 'ANDing' of two empty tray images. A tray loaded with the required number of empty items – a cup, two bowls, a plate, and a cutlery pack is then substituted at the inspection station.

The DUMC unit is loaded with the required software from the host development system and set running. The print option is set at the modify stage and the other parameters left at their default values.

The Apple then takes an image of the loaded tray, exclusive-ORs this with the ideal tray and transfers the resultant image to the DUMC unit. This image is then expanded as described and printed out on the dot-matrix printer. This print out is then examined and the count window corner coordinates of the shadows of interest checked against the default values. If they are different a note is made of the revised top left and lower right revised window corner coordinates from the printout. A number of runs is then made with the test tray disturbed each time and the correct windows set. The counts for each window noted and the average of each of these is then compared with the default threshold values. If, after subtracting about ten percent of the counts as tolerance, they are significantly different to the default the program is re-run with these new thresholds (less a given tolerance value). If the windows have been modified, the light source moved or the capture thresholds changed it is quite likely this will be necessary.

Another few runs of the system inspecting trays in different states of loading is used to confirm these thresholds which are modified again as required.

Once set up in this way the program then loops performing the required inspection on the various test images provided by the Apple with the state of loading being displayed on the DUMC unit VDU screen.

4.4.3 The Software Proposed for an Industrial Application.

This section contains a brief description of the way the software developed in the project would be applied in an industrial situation. The proposed software operation is as follows:-

1/ The Apple image grabbing 'front end' captures images whenever a tray arrives and tries to transmit these images to the DUMC system.

2/ The DUMC system cycles through its program sequence and accepts images when it has finished its last task and an image is made available by the Apple system.

The Apple system software would be quite similar to the development software producing images using preset ideal images and thresholds established during an initial set-up phase.

The DUMC unit software could dispense with much of the software in the above development stage after performing an initial setup stage which would establish the counting windows and their thresholds for the particular arrangement of test cell and object under test. The resulting program would then be blown onto eprom with these values preset as defaults. The operational program would only need to contain the following functions:-

1/ An interface function.

2/ An expand function.

3/ An area count function with no VDU output.

4/ A decision making function with a rejection indication.

The image print option would not be required and neither would any VDU screen indication at any stage. All that would be required would be some indication that the program was running for example by an LED on the unit, and some distinct external indication that a loading error had taken place in order to draw a supervisor's attention to the problem. This could be, for example, a flashing warning light or an audible alarm. The system would also need to provide a reset 'warm start' facility to restart the program operation after the correction of the problem.

CHAPTER 5

RESULTS FOR THE FIRST PROJECT.

This chapter contains the results obtained from the different stages of the development program. It is divided into the following sections:-

1/ The production of an 'ideal' image in the set-up phase.

2/ The Apple preprocessing stage.

3/ Establishing the DUMC Unit counting areas.

4/ A complete printout of an example DUMC Unit run.

5/ The results produced by full process test runs to establish count thresholds.

6/ A discussion of the results and an evaluation of the system performance.

The image representational figures were produced on a dot-matrix printer with a single dot per picture element (pixel), and with binary 'one' points being represented by a black dot. Note that the 'y' coordinates referred to increase down the image.

5.1 Producing an Ideal Image.

The set-up section of the Apple program produces the 'ideal' image. This is illustrated by taking the two separate typical images of the 'ideal' empty tray in figures 5.1 and 5.2 and logically 'ANDing' them together. This produces the 'ideal' image in figure 5.3. By comparison of the black areas highlighted in the figures it can be seen where the spurious boundary points, near the binary threshold, have been removed. This reduces digitzation errors in area counts in subsequent processing. This can be seen more clearly in figure 5.4 which is the result of 'EORing' the two images.

5.2 The Apple Pre-processing Stage.

Figure 5.5 shows an image of a typical correctly loaded tray and figure 5.6 shows the the resulting image when this is 'EORed' with the ideal image previously produced in the set-up phase. This image is now ready for transmission to the DUMC unit, with the differences in the two images highlighted.

5.3 The Setting of the Counting Area Limits.

Using the above printout in figure 5.6, produced on a printer connected to the DUMC Unit the count area divisions were taken as illustrated in figure 5.7. The important count area coordinates, –that is the coordinates of the top left and bottom right corners are then entered as defaults in the program. These are:–

TABLE 5.1.

Area no.	corresponding item	top left coords.	bottom right coords.
1	bowl	55, 0	71, 78
2	bowl	125, 0	156, 78
3	cup	233, 0	268, 48
4	plate	81, 87	110, 182
5	cultlery	253, 62	279, 168

5.4 A Complete DUMC Unit Run.

Figure 5.8 shows an example printout of the VDU output of one of the three count option variants of complete runs through of the development program, produced by connecting the VDU serial output line to the Epsom FX100 printer used to produce the dot-matrix images.

The different program stages are as follows :–

1/ Setting of the image print option.

2/ Setting of the count display option.

3/ Threshold modifying option.

4/ Count window modifying option.

5/ Greetings banner.

6/ Image expansion.

7/ Image printing.

8/ The selected count option display.

1/ The processed count.

2/ The processed count and area coordinates. (On printout).

3/ The invitation to modify count areas.

9/ The presence or absence of item indication.

The run time of the above program, with the given count windows, without the time consuming screen output routines, was in the region of 1 second.

5.5 Establishing the Count Thresholds.

The following table illustrates area counts produced by typical loaded tray images similar to those in figure 5.5. The first tray image is of an empty tray and the remainder are different images of correctly loaded trays.

TABLE 5.2.

Run number	figure number	Area number count				
		1	2	3	4	5
1 (empty)	5.9	0	9	14	0	7
2	5.10	392	587	1040	326	1700
3	5.11	466	589	1025	288	1721
4	5.12	413	586	990	324	2037
5	5.13	421	588	1016	318	1626
6	5.14	426	581	994	302	1979
7	5.15	405	580	1013	302	1142

From the minimums of these and subsequently measured values 10% was subtracted and the 'correctly loaded' thresholds indicated in table 5.3 were set.

TABLE 5.3.

	Area number				
	1	2	3	4	5
Minimums	364	559	990	295	1024
Thresholds used	328	503	891	266	922

5.6 The Results of Illustrative Full Test Runs.

The following table was produced by a series of test images, each of a tray incorrectly loaded in some aspect.

TABLE 5.4.

Item absent	figure number	Area counts				
		1	2	3	4	5
1	5.16	0	570	1029	303	1441
2	5.17	418	61	1019	325	1398
3	5.18	413	586	13	295	1427
4	5.19	364	587	1030	0	1412
5	5.20	421	582	1020	338	4
1 @ 2	5.21	0	62	1020	352	1665
1 2 3 5	5.22	0	15	11	349	0

Only items 1 and 2 were tested with both items absent together as these were the only items producing shadows that could possibly influence each other. Despite their closeness, however, it can be seen that the shadows have no effect on each other and that the difference falls well within the alloted 10% tolerance for them.

The terminal screen attached to the unit displayed in each case a message indicating the tray was incorrectly loaded and an indication of the area or areas in which the tray loading was in error.

The following table shows illustrative results produced by having the trays

under test with the particular loaded items badly misalligned within, or near to, their locating wells.

TABLE 5.5.

Items misaligned	figure number	Area counts				
		1	2	3	4	5
1	5.23a	298	611	1009	341	1024
1	5.23b	411	592	1023	338	1034
2	5.24a	434	845	1008	345	1034
2	5.24b	434	1042	1023	345	1020
3	5.25a	384	583	507	348	1032
3	5.25b	378	578	1126	345	1022
4	5.26a	442	587	1035	52	1296
4	5.26b	451	577	1003	274	1305
5	5.27a	400	559	964	299	2252
5	5.27b	400	559	907	300	2345

5.7 The Results Produced When Simulating Item Contents.

A further series of results were produced with the tray items being loaded with simulated contents for the projected application. These were strips of cream and pale pink card for the bowls and the plate, and the cream coloured milk carton for the cup. The results were as follows:–

TABLE 5.6.

Run number	figure number	Area number count				
		1	2	3	4	5
1	5.28	578	574	697	606	1196
2	5.29	400	572	966	650	1121
3	5.30	635	570	799	625	1138

5.8 Discussion of the Results.

Timings made on test runs show that the described algorithms produce the required window area counts in less than 1 second. This is an acceptible processing-time overhead and indicates the high level of computing efficiency achieved in the program.

The results of test runs in section 5.6 Table 5.4 show a large difference in the area counts for items present in their locating wells in comparison with when they are absent. The highest count of any area in which an item is absent can be seen to be less than 20% of the chosen minimum threshold for that area when correctly loaded. This count could be due to a number of factors including spurious lighting changes and digitization error. The low count in the case of items being absent suggests that the detection of the anticipated error condition of items not being loaded at all should be 100%.

If for some reason an item is missing and another placed exactly in its position to coincidentally produce a satisfactory count for that area, the fact that some other item would be missing from its locating well would still produce an error condition.

The results for misaligned items near to their respective locating wells in Table 5.5 show that the system cannot give a reliable indication of loading and loading error. In some cases the misalignment count falls below the error tolerance adjusted count and in other cases it is much higher. By considering the figures for misaligned items and the way the shadows of the misaligned items have moved it seems that by a finer tuning of the count areas and thresholds, detection of this misalignment could be accomplished. This may be necessary if the particular loading robot being used has poor repeatability or if the gripper is not releasing cleanly.

As mentioned in Chapter 4.4.1 the possibility of two items swapping place, or of spurious items being loaded onto the tray, or of items being stacked is considered negligibly low and is hence not tested for.

The results obtained in section 5.7, in which the item contents represent the prospective loadings in the given application, show that in most cases the count is significantly increased in comparison with the count for empty items. This demonstrates that the conditions in which the system has just been calibrated is the 'worst case' situation for detecting loading errors and that anything additional placed in

the tray items only makes the task easier. The only discrepancy to arise is in the unique case of the cup where the shadow count is made over the interior of the cup and in this case sections of the inner shadow are removed by the pale cream milk carton. In this specific case the window could be modified and the threshold adjusted correspondingly lower to allow for this.

5.9 Concluding Comments on the First Project.

The results have shown that the 'shadow EOR' method, devised to test for the range of anticipated robot loading errors in the proposed application, can be expected to produce reliable results for tray inspection. This is also achieved well within the limited image processing time available and suggests a hardware arrangement where a number of robot loading stations feed a single inspection station. More complex item shadows can be tested and verified by finer adjustment of search windows to include only those areas where shadows are cast and by making up compound windows out of a number of rectangular search areas. Because the technique is not specific to tray inspection it could be expected to work equally well in other applications where comparatively flat base objects are to be loaded in specific areas with opaque 3-D items of the same colour which stand above the surface.

CHAPTER 6

SECOND PROJECT

PROJECT OBJECTIVES AND METHOD OF APPROACH

6.1 Statement of Objective.

The eventual objective of the second project, which constitutes the major part of the work presented here, was to use a vision guided robot to automatically unload the recyclable items from a meal tray, on its return from use on an airliner. The contents of this tray are anticipated to be randomly arranged cups, bowls, and plates, containing discarded food and wrappers from prepackaged items, for example butter cartons, and salt, sugar, and pepper sachets.

The returning tray is submitted for inspection at an inspection station similar to the one in the first project. The four physical components of the robot cell system are:–

1/ A tray input roller-conveyor upon which the trays enter the cell and feed through it.

2/ An air jet station bridging the conveyor which blasts high pressure air across the incoming tray surface to remove light-weight debris.

3/ A visual inspection station bridging the conveyor which comprises a lightbox, camera and rotary table.

4/ A robot unloading section which comprises a tray locating section on the conveyor, a robot, and the bins into which the items to be recycled are placed by the robot.

This vision task can be considered a wholly 'robot vision' problem. The recyclable items of interest are the bowls, plates, cups, and cutlery. The vision system is used to provide identification and pick-up point information of these recyclable items on the tray to the robot further down the roller conveyor. The information is obtained by processing the tray image captured at the inspection station. The objective of the processing is to isolate significant areas of the image corresponding

to the recyclable items from a variety of remaining debris discarded in the eating process. The pick-up point coordinates would then be transformed into robot tool space coordinates. These and object identification information would be communicated to the robot, simultaneously with the tray being released to continue down the conveyor to the unloading station. It is assumed that with a sufficiently shallow conveyor gradient and gradual braking at the station that the items would not be disturbed in transit between these two points.

In the context of Chapter 1 in which the advantages of vision systems in general are discussed, to determine the feasability of implementing such a system in this second project, it is necessary to consider the way the task is performed at the moment.

At present the task of unloading the tray is split into a series of separate operations. For this reason it is not possible to make a direct comparison with this system. With the project in the development stage it is hard to see a low-cost vision-system-guided robot being able to match the performance of a practised human operator. The discrepancy in absolute speed of operation could be counteracted by the robot's continuous operation – with no necessity for meal breaks, or response to distractions. The vision system would require virtually no maintenance and the robot only service breaks at most every 300 hours of operation. As in the first project it was anticipated that the robot manipulation time would make up the bulk of the operating time so a processing time target of five seconds was taken.

In common with the first project the most important criterion for implementation of the system in an industrial environment would be its economic viability. This must again be assessed on the cost of the research and development undertaken and the cost of factory implementation as against potentially significant reductions in labour costs. As at present there are increasing labour costs and decreasing system costs this application seems to be becoming increasingly viable.

6.2 The Nature of the Problem.

This overall problem falls into the vision category of a bin-picking operation [32][33]. This is because the items being observed are not in fixed positions and may overlap and touch. It differs from the 'classical' bin-picking vision operation however, in that the items are not randomly orientated but instead are limited

in their 3-D orientations by their stable resting positions. Another dissimilarity is that unlike normal bin-picking operations in which a given item, for instance a particular casting, may only be obscured by similar items, the tray to be inspected contains a number of very different objects which may overlap. These items may also be obscured by additional random debris. When all these factors are considered together the problem becomes a complex one.

Because of this complexity it was recognised that a simplified system was needed to determine a working approach to the problem before a more generalized industrially implementable system could be developed. This system had to be able to operate on a subset of the overall problem. The simplified task chosen was the identification and orientation of the recyclable plates, cups and bowls but not cutlery, on a flat base tray. The items are clean and empty and in random orientations the correct way up, but not in close proximity to, or in physical contact with, other items on the tray.

To solve this simplified task the system had to be able to automatically differentiate tray items from a background of the same colour, irrespective of the item's orientation. In addition it also had to be able to determine the pick-up points of items within a given processing time of 5 seconds, and within a limited research budget, in order to make the system consistent with a low cost industrial application. It was also recognised from the outset that if the system were to be capable of some degree of intelligent unsupervised learning, its flexibility would be greatly increased. A further requirement was that it should be possible to extend the approach in future, to work on the problem of items such as a bowl or cup with contents, items in close proximity to other items, and items partially obscured by a variety of randomly orientated debris.

As in the first project, the most important requirement to dictate the choice of approach, was the need for the system to be able to differentiate a tray item from a background of the same colour. This 'worst case' situation arises when the recyclable items on a returning tray contain few or no identifiable contents and the number of features available to identify an item is severely limited. It is in consideration of the solution of this 'worst case' situation that the choice of approach to the simplified problem was made.

6.3 The Chosen Approach.

The approach adopted to the project task is represented schematically in figure 6.1.

Following the techniques developed in the first project, a shadow processing method was adopted for the second task. Thc image capture hardware was the same as in the first stage project, with additional attention being paid to the lighting arrangement of the tray. The software was developed as three separate programs. The image capture and transfer software was much the same as in the first project with a simplified Apple preprocessing stage. The DUMC M68020 system software was the same up to and including the image array expansion stage. The second project was mainly concerned with the development of suitable processing techniques to be applied to the expanded image to extract the required identification and orientation information.

In the following chapters the word 'item' is used to refer to a particular type of thing being inspected, for example a bowl, plate or cup. An 'object' or 'object class' is defined to be a subset of an item class for which the item is in a limited range of its possible orientations for which the measured characteristics have similar values. As a result of this definition a given item class, for all but the geometrically simplest of items, is made up of a number of object classes.

The approaches considered that could be used to determine the object classes of objects in the tray images fall roughly into the following categories:-

1/ **The Roster (or List) Method** which involves a comparison of the the test image with a list of all possible image patterns.

2/ **The Common Property Method** which involves a comparison of ranges of properties derived from items within the image with known object properties.

3/ **Clustering Methods** which involve using properties represented by real-valued features as coordinate components in an N-dimensional feature space. The objects can then be classified according to their geometrical relationships in this space.

The first project exemplifies the first of these categories. This approach can be used because a comparison can be made between the idealized template image and the test image as a whole. It is the presence of the discrepancy between these two images that is significant in determining the loading error and not the nature of the discrepancy.

The second and third techniques offer much greater versatility than this first method as they allow for variations in object properties within the image and can take into account the nature of the variations. The third approach which is essentially an extension of the second, allows established statistical techniques to be used in the recognition process. For these reasons it was decided to investigate automatic recognition techniques based on the last two of these approaches. It should be mentioned that industrial applications of cluster-based recognition techniques are rare and that for practical applications of such methods one has to look as far afield as biomedical and meteological applications [34][35].

Because of the great potential advantages offered by a fully automatic object learning, recognising and orientating system, approaches to automatic pattern recognition were considered. According to Tou and Gonzalez [36] these approaches fall into the following categories:–

1/ **Heuristic Methods** – based on human intuition and experience. These involve using a set of ad-hoc procedures for specifying tasks and their success depends largely on the knowledge and experience of the system designer.

2/ **Mathematical Methods** – based on classification rules formulated in a mathematical framework. These include common property and clustering concepts. There are two main divisions in this group the 'Deterministic' approach which is based upon iterative learning and the 'Statistical' approach which looks for optimal classification using statistical methods.

3/ **Linguistic Methods** – which are based on pattern classification by the use of primitive sub-patterns.

In the project research it was decided to investigate techniques based on the second of these approaches – the deterministic mathematical.

The subsequent chapters in the thesis describe the development steps to produce a working system based around this approach with chapters concerned with the separate processing steps applied to the image.

In Chapter 12 the performance of the developed routines is evaluated, first with images containing standard idealized shadows of items and then with mixed scenes of a variety of randomly arranged items. The chapter concludes with a resume of the objectives achieved with pointers suggesting the directions the work could explore in future developments.

CHAPTER 7

THE BEHAVIOUR OF SHADOWS

UNDER PROJECT LIGHTING CONDITIONS

This chapter describes the way shadows behave with respect to the illumination conditions and how they relate to the particular items casting them. An appreciation of this shadow behaviour is essential in understanding the methods used to deduce the object information from them.

7.1 The Lightbox Arrangement.

The inspection cell arrangement was similar to that used in the first project (figure 3.2), with the roller conveyor support for the tray under inspection as the tray passes through the cell. When the automated learning stage was introduced this roller conveyor support was replaced by a fixed tray with a stepper-motor driven circular centre section, at the normal tray inspection position. This rotary table is illustrated in figure 3.2a.

Shadows were again used in this second project to give a contrast between, and hence allow differentiation of, the cream coloured items on the tray from the cream coloured tray background. This means that the arrangement of the light source with respect to the tray was important when compared with the the light arrangement in the first project. A number of methods were investigated to produce a source of even, parallel light including using projector bulbs with lenses, drinking straws with standard bulbs to collimate the light and multiple light sources of variable intensities. Eventually it was found empirically that the original spot-lamp bulb at a sufficiently distant point, 8 feet above the rig, produced the most even, parallel illumination, despite the lower levels of light intensity, and hence shadow contrast at this distance.

The light source has three degrees of freedom with respect to the tray. These are the distance from the tray, the angle of elevation with respect to the tray, and the angle with respect to the line-of-centres in the plane of the tray. With the light source confined to move in a vertical plane its height dictates the shadow width

parallel to the incident light direction, the range of shadow width, and the amount of light shinning around the corners of the item. The height was chosen so that the light falls on the tray at a comparatively steep angle. This reduces the amount of corner contribution to the shadow, and produces a narrow, clearly defined shadow which decreases the likelihood of shadow regions overlapping or disturbing each other. The height chosen also had the effect of minimizing the amount of shadow width variation across the tray as a result of the increasing distance of items from the light source when moving from left to right across the tray.

The choice of light source was not crucial, and it was found that the 60 Watt reflector bulb used in the first project provided an adequate light intensity contrast for the digitizer between the shadow area and the cream tray surface even at a distance of 8 feet. In future developments with a less restricted tray arrangement, there may be dark items on the tray. In this case it may necessary to increase the light source intensity to give a better contrast between such dark items and the shadows cast by all items standing above the surface.

Another consideration with the light source was that the tray was evenly illuminated across its surface in order to use a fixed binary threshold. Using the spotlight the illumination across the central bright area created by the bulb was not of uniform brightness but contained brighter regions at its centre where there is a filament image. In order to produce the most even illumination of the tray the bulb was carefully directed at its centre. This helped to ensure that the edge of the tray nearest the light source and the edge furthest away are lit with the same brightness. With the spotlamp mounted in the overhead position the range of this brightness was small compared to that around shadows and hence caused few problems in setting the binary threshold.

A further consideration with respect to the lightbox arrangement was the apparent width of the shadows normal to the incident light direction as seen by the camera. If one considers the arrangement in figure 7.1 one can see that the amount of shadow visible to the camera depends on the x lateral position of the particular item on the tray.

The geometry of the arrangement gives:-

$$\tan A = \frac{CH}{XD} = \frac{IH}{Extwid}$$

$$Therefore \quad Extwid = \frac{(XD \times IH)}{CH}$$

Where CH = Camera height.

XD = x displacement of item from camera.

IH = Item height.

$Extwid$ = Change in apparent shadow width.

As the height of the item and the camera are constant it can be seen that the width of the shadow, and hence its area, is linearly dependent on its x displacement from the centre of the tray. This means that the measured width and area can be readily corrected when a true vertical overhead viewed value is required. This is necessary in the identification stage when an item anywhere on the tray must be compared to an item learnt whilst placed at the centre of the tray.

7.2 Considerations in Shadow Interpretation.

When considering techniques for shape interpretation, some of the qualities of the outlines of the shadows become significant. One such quality is that some outlines have re-entrant features. This is when a radius from a point on the interior, has a multiple crossing of the boundary. This is illustrated in figure 7.2. This quality is important in that it precludes some shape analysis techniques, for instance some Fourier transform techniques. These can be applied to line segment lengths, drawn at regular angular intervals, from some point in the interior to the perimeter. As this method relies on the function of radius length against angular displacement being continuous, if a radius has a multiple crossing of the perimeter the function is not continuous and consequently this method cannot be used.

The shadows produced by the plate, bowl, and cup have inner edges (that is the edges nearest the light source) that are more accurate representations of the shape of the item causing them than the outer edge. This inner edge is a region of higher contrast between the illumiated cream item and the dark shadow cast,

and is a region of almost step change in intensity. The rear edge of the shadow (furthest from the light source) is a more diffuse edge with a more gradual intensity gradient. This edge is more likely to be affected by scattered light and the shadows of other nearby items. For this reason a lower level of confidence should be placed in information derived from a rear shadow edge than that from a front edge.

7.3 Shadow Types.

A shadow type or object class contains a group of shadows, caused by a given tray item, in a range of orientations for which the chosen project identification features have similar values. This means that in the subsequent recognition process when a shadow is resolved into a given object class it has been interpreted as being of a certain orientation dependant shadow type.

These object classes can be illustrated by considering the case of the plate and bowl lying flat on the base tray the normal way up. Under these circumstances, by observing the shadows, one might intuitively make the assumption that there are three distinct shadow 'types'. These are cast by items in the following orientations:-

1/ The item's short or long axis aligned along the line of centres from the light source to the item centre. Refer to figures 7.3 and 7.3a for predictive scale drawings of this shadow and figure 12.7b for an actual bowl outline plot.

2/ One of the item's short corner pair of edge axes, nearly or fully aligned along the line of centres from the light source to the item. Refer to figures 7.4, 7.4a and 12.10b.

3/ Any other axial alignment of the item with respect to the light source. Refer to figure 12.8b.

The first case is anticipated to result in an inner edge of length equal to a single side length and two short corner lengths.

The second case is anticipated to result in two total side lengths, a short corner section between them with a negligible length contribution from another short corner side.

The third and most common class of shadows is a combination of one of each main side lengths, a short connecting corner length and some significant portion of a single short corner length at one end.

The cup illustrates the degree of complexity in the shape of shadows formed by a more irregular object. For cups lying flat on the tray the handle orientation causes large variations in shadow appearance and this appearance is greatly affected by the angle of the incident light source. For the angle of light source used in the project, if the handle is within 60° of the line of centres – light source to cup centre, on the light side, the handle makes no contribution to the main area of cup shadow. The shadow for these orientations is virtually rotationally invariant – compare the shadow outlines of cups in figures 12.23b to figure 12.26b, which are at 36° rotation intervals. If the cup handle is in angles from slightly before 100° to about 30° to the line of centres from light source to cup, on the side away from the light source, above and below the line, the handle makes a single contribution at one end of the shadow, for example figure 12.17b and 12.18b. Below about 30° to this line the handle causes the shadow to be split into two sections, significantly affecting the shadow appearance – figures 12.19b and 12.20b.

The examples given here of different object classes have been intuitively reasoned by observing a series of images containing the items over a range of anticpated orientations. The way such decisions are made in an automatic, unsupervised recognition system depends on the nature of that system, for instance whether the system is a heuristic, 2nd generation system, or a cluster based, adaptive learning, 3rd generation system. The final approach was the one developed in the second vision project. The resulting object classes produced by this approach are illustrated in later chapters and make for interesting comparison with the anticipated object classes proposed here.

In the long term project objective, beyond the simplified task, a more involved problem arises when shadows of nearby shapes combine to produce complex compound shadows. The use of multiple light sources switched on in sequence, with different images taken at each switching, would perhaps offer a possible solution to such problems. For the simplified application however, the processing time penalty was considered prohibitive because of the multiplication of images to be processed. In the long term it is intended that these compound shadow problems will be solved at the software level. This may well involve making greater use of shadow properties

like the property that any inner shadow edge produced by an anticipated item has only one single concave region and that any shadow edge containing more than this one region represents a compound shadow and should be segmented at the ends of the concave regions.

In the simplified problem it is anticipated that features derived from the inside edge, outside edge or other boundary information will be sufficiently distinct to allow the tray items to be distinguished and orientated. The necessary information is certainly all available in the shadow outlines. Hence the feasibility of the project, in terms of accomplishing the project objectives, is dependent on the amount of computing effort, and hence processing time, that is required to extract and process this information.

CHAPTER 8

PRELIMINARY IMAGE PROCESSING.

This chapter contains a description of the primary processing operations that were applied to the tray image to facilitate the subsequent extraction of features. They are mainly concerned with image thinning/edge-extraction and outline tracking. Following these descriptions there is a discussion of the alternative techniques that were considered, and the reasons for the preferred choice of approach.

8.1 The Image Capture and Transfer Software.

The Apple image-capture software was the same as described in the first section of the thesis (section 4.4.2.1). Subsequent Apple processing differed in that instead of the image being exclusive-ORed with another image the raw binary image was transferred across to the DUMC unit by the high speed parallel link.

The DUMC M68020 unit image reception software and the subequent image expansion software, in which an image array is expanded in to an array in which one byte corresponds to one pixel was the same as in the first project (sections 4.4.2.2 and 4.4.2.3).

8.2 The Image Thinning/Edge Extraction Algorithms.

In order to simplify the subsequent tracking routine and retain the modular structure of the program the next level of processing applied to the expanded binary image was a combined thinning/edge extraction algorithm. This produced a binary skeleton image in which the edges of shadows in the original image were set to a digital 'one', and the remainder of the image set to digital zero. The requirements of this process are that the resulting edge image information remains faithful to the original shadow image information and the operation is performed rapidly in real time. The requirement that the edge remains faithful means that connected edge points in the original image remain connected in the skeleton image, and that there must be no significant loss or corruption of image information.

In the standard approach to edge detection the image is regarded as ideally composed of 'constant' regions separated by step edges [37][38]. Under normal circumstances in a more typical vision scene, for example in an image of a house, this assumption is an over simplification. In this case a more generalized approach is needed which detects intensity edges resulting from different kinds of scene discontinuity, including shadow edges, slope edges, range edges, and texture edges.

Much of the information required for this more generalized approach is unavailable from a binary system and can only be provided by a grey level system [39][40]. As however, the particular image being worked on was a very limited variant of the general scene case, in which the edges around shadows can be considered as step edges, the process of edge extraction was considerably simplified.

The three algorithms developed had the effect of generating an 'edge' either around the outside or inside of a given shadow. The 'edge' generated by these algorithms is in accordance with the 'border' as defined by Rozenfeld [41]. He places the 'edge' in a binary image half way between the bright and dark pixels on the border. As the anticipated resolution required of the image system was as low as 10mm. the $\frac{1}{2}$ - 1 mm. difference this distinction makes was not seen as significant and in this thesis the word 'edge' is used synonymously with the word 'border'. All three algorithms operate on the four-connected immediate neighbours of a point and are based on similar principles but differ significantly in detail.

In the following description a pixel in condition digital one, representing a bright area of the image, is described as being 'on', and a pixel in condition digital zero, representing a dark area in the image, is described as being 'off'. Refer to figure 8.1 for a schematic diagram illustrating the behaviour of the following algorithms.

The first algorithm developed (appendix AP1.3.2.1.6) involved scanning the image to find 'on' points with at least one of the surrounding four-connected points 'off'. Only these 'on' points were allowed to stay 'on' in the final image, with all the other points originally 'on' being changed to 'off' points. This algorithm was implemented in 'C' and found to provide good continuous outlines in most discrete shadow conditions. These outlines were found to be satisfactory in the case of the restricted development project objective in which only separate tray objects were considered (results Chapter 12.2.1). Looking ahead however, to the to the extended problem of nearby or overlapping objects, the approach has the disadvantage that, in the case of two distinct but nearby shadows a shadow skeleton could be produced

that would lead to the linking of the two shadows at the subsequent tracking stage. This is illustrated in figure 8.1 by the common outline points on the two shadows. To combat this problem another algorithm was developed.

First it was decided to simply build up a shadow outline around the inside of the shadow by a similar method to the one above. This was accomplished by scanning the image array until an 'off' point was found. The four-connected neighbours of this point were then scanned and if at least one found to be 'on' the central 'off' point was switched on in the final image. The final image was made up of only these points set 'on' (figure 8.1).

Although this algorithm had the effect of maintaining separate shadows it produced frequent single-point-width shadow outline sections which acted as 'blind alleys' to the tracking routines subsequently applied to the image (results 12.2.1). These tracking routines will not 'back-track' along their original path and hence the resulting outlines produced in such cases were disjointed and complicated further processing. For this reason a more involved edge extraction algorithm was developed (appendix AP1.3.2.1.6).

In this alternative algorithm the image is scanned until an 'off' point is found. The four-connected points are scanned in sequence and noted. A set of logical conditions is then applied to these:-

1/ If all four are off, the centre point stays off.

2/ If all four are on, the centre point stays off.

3/ If two opposing points are on (and two off),the point stays off.

4/ If just one point is on, or just one point off, the centre point is switched on.

This choice of rules had the effect of maintaining separate shadows whilst removing the narrow outline sections of single pixel width. Although this removal contravenes the initial premise of preserving the connectivity of points in the resultant image, it was confirmed in test image processing that only insignificant line sections are removed (figure 8.1). It was also shown that with the subsequent application of the particular tracking algorithm to the edge-extracted image continuous outlines were re-established (results Chapter 12.2.1).

The second and third algorithms apply processing to 'off' points. Because

there are far fewer 'off' than 'on' points in the original image, in addition to the programming of these algorithms in assembly language, a significant processing time saving was made in comparison with the first method.

All the above algorithms are essentially parallel operations. The result of an operation on one pixel has no effect on the neighbouring pixels unlike, for instance, in 'blob growing techniques' [42]. For this reason the technique lends itself to a hardware implementation which would offer significant savings in processing time. Because of this parallel nature all three approaches involve a process equivalent to keeping a second modified image array along side the original. This was achieved in the software by labelling points as numbers other than logical one in the expanded image array as illustrated in the program section of the appendix (AP1.3.2.1.6).

The approach described yielded the desired continuous outlines in a simple way that is economical in its use of computing time. For comparison there is a description of more sophisticated techniques involving averaging and thinning operations in the discussion at the end of this chapter. In the context of the given application however, they were considered unnecessarily involved and suffer from an excessive processing time cost when implemented on conventional sequential hardware.

8.3 The Tracking Technique.

The stage of processing directly following the above thinning operation was a tracking algorithm. This broke down the edge-extracted image into a series of chain-coded lines suitable for further processing. The tracking technique applied to the edge extracted image was a standard Freeman chain coding technique [43]. The simplicity of this implementation is due to already having extracted the shadow outline at the previous stage. If this were not the case and the raw binary image was being processed, significantly more processing, involving considering the state and relationship of a much larger number of neighbouring pixels, would have been necessary. The algorithm works as follows:–

Each absolute direction about a point is given a label as indicated:-

8 1 2

7 P 3

6 5 4

The direction of track is initialized to 5. The image is then scanned from the top left corner until a digital 'one' point is reached. This point is labelled and stored in an array, and the corresponding pixel labelled 2 in the image array so as to prevent its retracking. This point is then taken as the starting point of a line segment and stored in a start point array. A series of points is then built up by searching in as near the original direction as possible to find the next point. This is accomplished by looking in the search direction taken to find the current point and then if no point is found in this direction 45° either side and so on at 45° intervals up to 135° . Each time a point is found it is stored in the point array and labelled to remove it from the image array. If no further continuation point is found the tracking procedure terminates; an end character is put into the point array and the final point is put into an endpoint array. The image is then scanned, from the point after the previous line starting point, for the next non-tracked image point. The procedure is repeated until the whole array has been processed and all the image broken down into a series of lines of connected points (appendix AP1.3.2.1.7).

8.4 Alternative Averaging and Edge Extraction Algorithms.

Often in image processing the preprocessing operation includes an initial averaging or smoothing stage in order to homogenize the outlines. This involves bridging gaps, eliminating voids and removing spurious noise, before the edge extraction algorithm is applied. In the image being worked on, because of the high contrast between shadow and bright background resulting from the application of structured lighting, there were very few spurious irregularities in the shadow outlines or other spurious points in the scene. As a consequence no averaging operation was considered necessary. The potential averaging techniques considered were similar in approach to those first described by Dineen [44]. He proposed the use of different sized windows about a central pixel, typically 7 × 7 pixels, and depending whether

the count of 'on' pixels in this window is above an arbitrary threshold the central pixel is switched on or off. If this threshold is low a thickening of an image takes place and if high a thinning takes place. Under normal circumstances of an image in which features are made logical one, low thresholds smooth out irregularities and high thresholds emphasize them.

Alternative methods are proposed by C. Archelli et al. [45], and Unger [46] for averaging using 3×3 elements. The methods use the logical relationships of points within the chosen window.

With regard to binary edging operations Dineen, in early image processing, proposed an edging technique which effectively involved preserving the centres of asymmetry. It works by operating on windows of seven units by seven. The method first finds a pixel which is 'on' and places the 7×7 cell about this point. The elements about it are then scanned until another 'on' point is found. The three diagonally opposite pixels are considered and if they are all zero the count is increased by one. The process is repeated until no further 'on' points are found. If the total count is greater than a given threshold then the centre pixel remains on. The threshold can be set automatically to take into account the local light intensity by counting up the total number of window pixels 'on' and taking some proportion of this as the threshold. In the case of both the smoothing algorithm and the alternative edge-detection algorithm, even operating on smaller windows of 5×5 or 3×3 pixels, the marginal improvement in algorithm performance anticipated would have been offset by an unacceptible increase in computing time overhead. For this reason the algorithms were not considered viable alternatives to the ones adopted in section 8.2.

CHAPTER 9

IMAGE FEATURE EXTRACTION

As described in Chapter 6 the approach chosen for image object recognition is based upon making a comparison of the properties of objects in the test image with the properties of objects already known to the system. In order to relate the 'bottom-up' image information to the 'top-down' prior knowledge it is necessary to have a common way of expressing these properties. This is achieved by representing them by a range of image features. This chapter contains a discussion of the nature of these features and the factors involved in choosing satisfactory ones for general applications. This is followed by a description of the features chosen for the given application and the way they are derived. Next the factors which control the stage at which such features are extracted within a recognition process are briefly discussed and the chapter concludes with a description of the feature extraction structure adopted for the project.

9.1 The Nature of Features.

The stages following the image preprocessing in the recognition process involve the extraction of features from significant chain-coded tracked outlines and using these to resolve outlines into the particular object classes giving rise to them.

In generalized feature-based pattern recognition significant points along the outlines or general properties of the outline are taken and real-valued features such as area, or corner frequency characteristics are derived from them. The result of this resolution of an outline into features is to produce a vector representation of an object in an N-dimensional feature space. For example a feature vector defining an object in feature space can take the form of the following column vector:-

$$x^* = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ \cdots \\ x_n \end{pmatrix}$$

Where $x_1....x_n$ correspond to different features. The treatment of these vectors to produce object classes and achieve object recognition is described in Chapter 10 on object recognition and discrimination.

9.2 The Criteria for the Choice of Features.

In order to reduce the complexity of the functions needed to determine features and the complexity of a subsequent decision function, a number of conditions are usually applied to the choice of features for pattern recognition. These are as follows in approximate order of importance:-

1/ Orientation invariance.

2/ Scale invariance.

3/ Uniqueness.

4/ Ease of feature extraction.

5/ Ease of physical interpretation.

9.2.1 Orientation invariance.

Few image features are orientation invariant in three dimensions. These include colour and selected dimensions of a limited subset of solid shapes such as spheres. In the majority of current vision system applications, images are simplified to be two dimensional, for example in the automatic interpretation of aerial photographs. The possible three-dimensional orientations of a number of objects is limited by the restricted number of stable resting positions for these objects. If each of these stable resting positions is considered to produce a different object in the image the identification problem is reduced to a two dimensional one in which one can make use of an object's rotationally invariant features. This approach is adopted in the identification of grounded aircraft in aerial photographs [47][25]. In developing the technique for this project, similar assumptions were made with respect to the number of stable resting positions of the particular items on the tray. This considerably reduces the number of possible shadow outlines to be identified.

Typical rotationally invariant features of an object are line lengths of certain significant lines, for example maximum radius from centroid, its perimeter length, its area, and its colour. Angles of particular features, for example internal holes

or specific corners, can be considered invariant when taken with respect to a fixed internal feature such as the maximum radius from centroid to the perimeter and not with respect to the image axis.

9.2.2 Scale invariance.

Scale invariance is the requirement that a feature is object size independant. This means it does not change with the size of the object or with the camera-object distance. This is important when observing a generalized scene where the camera-object distance is not known or where an auto-zooming technique is used with the zoom adjustment not known.

Typical features in this category are angles between two distinct lines, and the number of corners of a shape. Often shapes are made artificially scale invariant by normalizing them with respect to some unique dimension, such as the length of a unique maximum radius from the centroid.

In the particular application there were a limited number of reliable features available to distinguish between shadows of different object classes. For this reason, with a fixed camera-object distance, measured dimensions were used as identification features in their own right.

9.2.3 Uniqueness.

For the purposes of identification, in an ideal situation, the derived features should facilitate differentiation between any two object classes. If for instance, the feature control structure is based on a decision tree which distinguishes items into object classes, the more unique a feature is to one class the more quickly the tree diverges and an identification made.

9.2.4 Ease of feature extraction.

The degree of programming complexity required to calculate a feature is another important factor in determining the processing time cost of object identification. If a number of features must be calculated for each shape in a scene containing many separate objects, the processing time can be significant. For example with a Freeman chain-coded line, it is a comparatively easy task to calculate perimeter length and shape area for a closed outline. In contrast the calculation

of higher moments of inertia [47] is a computationally involved process, and the computation time penalty in deriving these must be considered in relation to the value of the information obtained.

In this particular application the chain-coding technique does not necessarily provide a closed shadow outline. If the outline is to be used to provide certain features accurately, for example shadow area, it is essential that it is closed. The computational penalty of closing the line artificially must be taken into account when considering the overall computational cost of the feature.

9.2.5 Ease of physical interpretation.

In assessing which features are to be chosen for identification it is easier, though not essential, for the programmer if the features can be easily physically interpreted. Examples of this are the length of maximum radius from a centroid or the area enclosed by an outline. If a quantity such as the Fourier series coefficients of the function of radius lengths from centroid to perimeter are used, the programmer has more difficulty in visualizing the solution, which makes it more difficult to program intuitively.

9.3 The Features Chosen for the Project.

Having considered the above criteria, combinations of a set of relatively simple features were chosen to make up the feature vectors in the particular application. These were:-

1/ The heuristic compaction factor, based on the enclosed shadow area and the closed shadow perimeter length.

2, 3, 4/ Three inside shadow edge lengths between conditionally defined points on the first concave region of the shadow.
(The inside edge of the shadow is defined to be the long edge nearest the light source). These are:-
the minpoint-to-maxpoint distance (d(min-max)),
the minpoint-to-midpoint distance (d(min-mid)),
and the midpoint-to-maxpoint distance (d(mid-max)).
These points are defined in section 9.4.2.

In addition, for the system to be able to determine pick-up points as well as recognise an object automatically, three pseudo features were derived in the object learning stage. These were distance measurements made between the three conditionally defined points mentioned in 2, 3, and 4 above, and the centre of a matt black circular pick-up marker placed on an item in the position where it was to be picked-up (figure 9.1). Although these features were not used in object recognition, they were used as one factor in the control of the automatic generation of object classes using the other features. A set of these pseudo features was then associated with each resultant object class for pick-up purposes.

As mentioned in Chapter 7 the chosen features are not continuous functions of an item's angle of rotation. The function of feature size against item rotation contains discontinuities at certain angles for different features for some items. These discontinuities result from the combined effects of item edge alignment with the incident light direction, digitization effects due to the digitized image resolution and failings in the tracking algorithm to produce continuous, closed outlines. These effects can be illustrated by considering the plots of theoretical feature values, derived from geometric functions, and actual measured feature values, against angles of rotation item rotation, for rectangular box shapes with the same dimensions as the bowls within the project. Theoretical plots for perimeter and area can be seen in figures 9.2 and 9.3 respectively, with comparitive plots of actual measured features in figures 9.4 and 9.5 respectively.

The computer program written to generate these features theoretically is described in appendix AP1.4.3.1. It works by considering an ideal parallel plane light source to be incident at an angle to the horizontal on an opaque item. Each point on the top edge of the item is considered to produce a shadow edge point a distance from the base of the item determined by the height of the item and the elevation of the light source (figure 9.6). In addition the program has a function which tests to see how closely aligned an item's edge is with the incident light direction. When this angle was below a certain limit the function tried to approximate digitization error due to the finite image pixel size.

For the relatively simple shape of the bowl and plate items it is possible to predict the shadow behaviour approximately using projection geometry as above. This was not seen as a satisfactory way of predicting the shadow properties of more complex shapes. If the cup, which is still a comparitively regular shape, had to

be modelled the function to do so would be considerably more involved than the simple function generated for the bowl. This approach would have also required a computer-aided-design (CAD) type interface and a skilled operator to input each particular new item into the system. For this reason it was decided to use features 'learned' from actual items in their various orientations in an object 'learning' stage.

9.4 The Feature Extraction Software.

Three different methods were used to obtain the four identification and three pick-up features chosen in the project. The heuristic compaction factor derived from the division of the squared, closed perimeter length by the shadow area (P^2/A) was provided by a variant of a standard chain-coded line processing technique [48]. The three features based on three specific conditionally defined points on the outline were found by a different method in which conditions were applied to points along the inside edge of the chain-coded outlines. The three pick-up features made use of these three points and in addition the software determined the centre of a black pick-up point marker and calculated distances to this.

9.4.1 Chain-coded feature derivation.

Refer to figure 9.7.

In order to calculate the compaction factor it was first necessary to determine independantly the shadow perimeter length and the enclosed shadow area.

The perimeter length was only calculated if the boundary represented by the coded line was found to be closed. Only in this case was the feature meaningful.

In order to determine whether a line was closed a small function *fclosed()* (appendix AP1.3.2.1.8) was run which tested whether the final point on the line was within a certain x and y displacement of the first point. If it was the function set a global marker to indicate this for the given line. The perimeter routine *fperi()* (appendix AP1.3.2.1.11) first checked this marker. If it was set the routine proceeded to use the information in the direction coded line elements. As the point directions are labelled from 1 to 8 in a clockwise direction from the vertical at 45° intervals, any odd elements are considered to contribute a single unit to the length and any even elements square-root of 2 units to the length (figure 9.7).

$$\text{perimeter length} = (\text{Even} \times \sqrt{2}) + (\text{Odd})$$

In the function the number of odd and even elements was first summed, the contribution of each determined, and the total length calculated by summing these. The resultant length was then stored as with the number of boundary points in a globally accessible line number indexed array.

A standard chain-coded technique was also used to determine the shadow area. For the area to be a meaningful feature it is also necessary that the line be closed. In addition to this requirement however, in order to calculate an accurate value of a shadow's area more stringent conditions must be applied to the start and end point of the line.

The function to determine shadow area (*farea()* AP1.3.2.1.10) worked by calculating vertical area sections between the horizontal axis and each given point following the outline around the shape (figure 9.7). The sections are arbitrarily chosen to make positive contributions to the area when x is increasing and negative contributions when x is decreasing. No area contribution is made when the line moves in a vertical direction. The modulus of the resultant area calculated in this way was then taken as the shadow area.

It can be seen that if the line does not start and finish with the same x coordinate the resultant calculated area could be significantly wrong by a number of columns. For instance if the shadow starting point is one third of the way down the image and the start and finish points differ by three x points the error is given by:–

$$\frac{192}{3} \times 3 = 192 \quad \text{pixels.}$$

In a typical area of 1000 pixels this represents a possible 20% error. For this reason in addition to the closed test another function *xend()* (appendix AP1.3.2.1.9) was used which equalises the start and end point x-coordinates. This was accomplished by taking the end with the smallest x-coordinate and following it back along the line until a point was reached which had the same x-coordinate as the other end. This provided a new alternative start or finish to the line. In doing so it was assumed that because of the narrowing nature of the end of the shadow outline only a minimal amount of enclosed area was lost.

Having developed this end equalization function the area function produced considerably more consistent results using these modified endpoints. The areas were stored in globally accessible line number indexed arrays.

The function *compact()* (appendix AP1.3.2.1.12) produced the compaction factor directly by squaring the global perimeter array element and dividing the result by the equivalent global area array element for each outline section above a threshold length. The result was also stored in a globally accessible array.

$$\text{Compaction factor} \quad C = P^2/A$$

Of the four identification features used this feature was the only one to give height information about an object. This is due to the shadow area and to a lesser extent the shadow perimeter being related to an object's height. This makes this feature important in distinguishing certain classes of objects, for example upright cylinders from disks.

The compaction factor feature is dependent on the width of the shadow cast in the same direction as the line of centres – light source to tray. Because of this, the feature is sensitive to the angle of elevation of the light source, and its distance from the tray and can vary across the tray surface. It was found that with a well adjusted light-source-tray arrangement the effect of shadow width variation could be minimized.

9.4.2 The alternative feature derivations.

The three alternative features used have the advantage that they can be calculated for non-closed outlines. It is anticipated that in extending the technique to operate on overlapping or touching shapes, this method would be more likely to extract meaningful features than the standard chain coding feature methods.

The features are based on distances between three conditional points along the edge of the shadow – the 'minpoint', the 'midpoint', and the 'maxpoint' defined as follows:–

Refer to figure 9.8.

The **minpoint** is a point on the inside edge that has the lowest y coordinate(y increasing downwards) along the line with both the x

and y coordinates of the subsequent point being greater than at the given point and with the five subsequent points to this having both x and y coordinates greater than or equal to the preceeding point.

The midpoint is the point on the inside edge of the first concave region, between the minpoint and maxpoint, which has the maximum x coordinate with a point following with greater or equal x coordinate and a point following that with a point with a greater or equal x coordinate than the preceeding point, and the lowest y coordinate value for that minpoint x.

The maxpoint is the point at the bottom of the first concave region on the inside edge with the greatest y coordinate that is preceded by four points with greater or equal x coordinates and less than or equal y coordinates and followed by a point whose x coordinate is less than the x-coordinate of the chosen point, with the point after this having an x coordinate less than or equal to that at the maxpoint.

The sequence to calculate these points was as follows. The outline was first checked to see if it was closed. Then based on the definitions of the three points minpoint, midpoint, and maxpoint, the function *shadw()* (appendix AP1.3.2.1.13) determined the coordinates of the three points for each line section above the threshold length and stored them in a globally accessible point array. At the program development stage these points were used to provide a whole series of features in the function *wlcimp()* (appendix AP1.3.2.1.14), for example a series of differences in x and y coordinates between them. A printing subroutine *printable()* (appendix AP1.4.1.4) was written which gave the user the option of printing out the complete range of features for any lines in an image in a tabular form. This made it considerably more easy to compare features and decide on the features most satisfactory for distinguishing between objects.

As described earlier the three features that were found to be most valuable were the diagonal distance between the minpoint and maxpoint, the distance from the minpoint to midpoint and the distance from the midpoint to maxpoint (figure 9.8). The first of these features was found to be highly reliable and consistent in discriminating between objects. The other two were important in providing orien-

tation information about objects in a scene. For example to provide an indication of whether a bowl was in a rotation angle between 0° and 90° or 90° and 180° . The importance of these features can be seen in figure 9.9 and figure 9.10 which are plots of these features actually measured for a bowl. Whereas the compaction and minmax distance features would be expected to be symmetrical for a bowl above and below 90° it is clear that these features are asymmetrical.

The limit of four chosen features was imposed by the large memory and high speed requirements of later recognition processing, despite the fact that more features would have inevitably improved recognition reliability. The choice of which features to use was made intuitively. This would perhaps offer an interesting application for covariance analysis to make the optimum choice of features to give the best object discrimination although this is complicated by not knowing the object classes in advance.

9.4.3 Pick-up point feature derivation.

These pick-up point pseudo features were measurements made on items in a learning stage only (figure 9.1). In this stage a circular matt black pick-up point marker was placed on an item where it was to be picked up. The circle was tangential to the line from the camera to the tray surface to ensure that the circle retained its circular appearance and could be identified as such by the vision system. The black marker was treated the same as a shadow by the thinning and tracking algorithms. Once the three endpoints of the true item shadow were located as described above, the routine *ccircle()* (AP1.3.2.1.15) finds the associated pick-up point marker and calculates its centre. The three distance features DtoMin, DtoMid, and DtoMax, the distances of the centre of the circle from the three points was then calculated in the function *dcentre()* (AP1.3.2.1.16). These features were also stored in a line indexed array.

9.5 The Feature Extraction Control Structure.

In addition to the factors that need to be considered when choosing features for an application another important consideration is the control structure of the program. This determines the order in which and the stage at which features are extracted to facilitate test object identification [49][50][51].

The problem of choosing a control structure for feature analysis has attracted a considerable amount of attention in image processing work. Some general factors to be considered when deciding on a suitable one for a specific application are:-

1/ The complexity of the the scene to be analysed.

2/ The type of feature approach adopted.

3/ The accessibility of features within the scene.

4/ Access to dedicated specialized hardware.

5/ The computing time available to accomplish the given task.

6/ Any financial limitations placed on the project.

9.5.1 Scene complexity.

This is highly dependant on the task being undertaken. In generalized scene analysis, which falls under our definition of computer vision, the requirement is for a system more analagous to the human vision system. The control system development in such cases has consequently been complex and involved an investigation of the relationship between information gathered 'bottom-up' and 'top-down'. 'Bottom-up' is that information derived wholly from the scene; for example shape, texture, and colour boundaries and their properties. 'Top-down' information is that information due to prior contextual knowledge of the anticipated scene [52][53][54][26][55]. The problem is related to incorporating syntactic and contextual information into natural language processing.

Some vision control structures, based around natural language processing, have been developed specifically for robotics applications [56]. Most of the control structures devised around this combined approach have been in artificial intelligence fields. This is an inevitable consequence of the relative importance we as humans place on such vision skills. The foundation for these studies is knowledge based interpretation and understanding [24][49][50][51][57].

In the majority of industrial vision applications a scene is severely restricted or confined to vary only in certain known respects. For these applications a much simpler control structure can be used in which a heavy dependence is placed on the 'top down' approach. Features in these cases are selected to verify or conform to 'top down' knowledge rather than being looked at in order to contribute to an

overall interpretation of the scene.

9.5.2 Accessibility of features / Type of feature approach adopted.

In an image where a large number of global features, for example object area or moment of inertia, are readily available, which provide the information necessary to discriminate an object, the feature control structure can be made comparatively simple. In some cases however, the global feature information is insufficient to make this discrimination. This can be the case when the features are unavailable, for instance when objects overlap or touch, or if only a partial view of an object is available. Under these circumstances one is restricted to operating on extended or local features. Typical extended features include large segments of an object's boundary, and textural information whereas typical local features are small holes and corners. Often more involved control structures are needed for approaches based on these two different types of features.

Approaches based on extended features depend on the quality of the features. This is in terms of accuracy, consistancy, reliability of extraction and uniqueness. If the features are unique enough in themselves – as is the case with some of the shadow inner edge line segments in the project, they can provide sufficient information to give the required object discrimination. Often however, additional processing is needed to link these together to form global features. An instance of this is when separate line segments are combined sequentially to produce continuous 'closed' object outlines.

Local feature methods are often employed in 2D cases of images containing overlapping or touching objects in which only a small set of an object's features are available. In the Local Feature Focus method [58] the control system locates and extracts a number of local features. One of these features, which is known to be reliable, is taken as a focus. The object is hypothesized around this and its neighbouring features using a maximal clique technique [59]. This determines the object orientation that is most consistant with the arrangement of the neighbouring features. The hypothesised object is then verified by reference to more remote local features. The identity of this method can be attributed entirely to the feature processing control structure as the method has no dependence on the nature of the local features or the way in which the features are derived. Similar approaches to the interpretation of 3D images have been made by Goad in [60] and Bolles in [32].

9.5.3 Access to specialized hardware / Time and financial limitations.

These factors are inevitably interelated. If a complex task must be accomplished in a relatively short period of time it may well be that the only way to do so is by using special purpose hardware, for example parallel processors, systolic 'bit sliced' processors [61], and purpose designed graphics processors like the Texas instruments TMS34010 [62]. This specialized hardware is often too expensive for a small research project working within a modest budget, or cannot be financially justified in a small scale industrial application.

If special purpose hardware is unavailable the time factor on a conventional serial hardware system is often a prime consideration. This has already been emphasised in the section describing the priorities involved in extracting individual features. In addition however to the value of individual features it is the control system which determines when features are deduced and in which order. The organisation of this structure can have a significant effect on the total image processing time.

9.5.4 Types of feature extraction control structures.

There are large differences in approach to feature extraction control structures, both in complexity and scope of application. In some cases, because sufficient time is available, a simple control structure can be implemented. In this case the values of all the possible chosen features are calculated at a feature extraction stage and any redundancy in the feature information in a subsequent recognition stage is ignored. Alternatively the control system can include 'data-driven elements' in which a small number of initial features are calculated first using a-priori information about the scene. These are then used to direct subsequent feature extraction through a series of similar steps until the required object identification and orientation can be made. This method has the advantage of only finding those features required for the identification. In applications where a varied range of objects is under inspection this could offer major time savings. This would be especially so in a case where some of the objects are easily discernible with a minimum number of features and others require the derivation of a large number of computationally complex features before an identification can be made. The penalty incurred by this type of control system is in the complexity of programming. It can often involve

the passing of control between many different levels within the processing program, and possibly includes recursion. This makes the tracking and debugging of such a program a difficult task [51].

9.5.5 The particular application in this project.

With regard to scene complexity the eventual objective of the second project falls roughly into the category of a 'bin picking' operation. This is in the sense that items to be picked up may overlap and touch. It differs however, from the classical 'bin picking' vision operation in that the objects are not randomly orientated but instead are limited in their 3-D orientations by their stable resting positions. They are also of differing types and have random debris associated with them. To achieve this long term project objective, working with a complex scene, a more versatile control structure would be required. In the case however of a scene restricted to a limited set of separate recyclable items, flat on the base tray, the problem is of a more simple industrial inspection type. The items within the scene are inspected with respect to simple 'top down' models and a much less involved control structure can be used to accomplish this task.

The possibility of having a data-driven hierarchical feature control structure was considered. This approach would have used a priority based feature selection structure to extract features as required, to achieve object identification. The approach would also have possibly involved additional functions to resegment line objects so that new features could be derived from these new line objects. More use of overall knowledge about the scene was also considered. For instance if it is known that only a single cup will be present, after positively identifying and locating a cup the subsequent processing would no longer need to include a search for a cup. These alternative approaches were not pursued however, due to the fact that they were viewed as unnecessarily involved and potentially time consuming for the comparatively simple feature extraction task.

The recognition method developed uses only the global and extended features chosen to identify objects. The lack of well defined local features, (no holes and poorly defined corners), precluded the use of a local feature focus method [58]. With the speed of the DUMC M68020 unit and the programming efficiency achieved in the earlier program stages, for example in the thinning/outline extraction and tracking algorithms, and with the clear shape distinctions possible by using well chosen

features, there was sufficient processing time available for a more simple feature processing control structure. For this reason, in the mathematical deterministic approach adopted, the four selected features were calculated at the outset of the learning and recognition programs for each object which was represented by a line section of length greater than a minimum threshold. This feature information was then passed to subsequent cluster-based object-class-determination and recognition and orientation stages which are described in the following chapter.

CHAPTER 10

OBJECT RECOGNITION AND DISCRIMINATION

After the mechanisms to produce image features have been developed, two stages of processing are necessary to recognise the objects within an image. The first stage involves building up a 'library' of objects to be identified in terms of the features. The second stage is in the image inspection program when a set of features derived from an outline in an image must be associated with the corresponding library object.

The way the library of object classes is built up and how test features are referred to it depends on the recognition approach adopted. As stated in Chapter 6, the approach taken in the project is based on mathematical determinist concepts, and is intended to produce a system requiring a minimum of skilled operator supervision. It uses statistical mathematical principles to divide up the chosen four dimensional feature space into object clusters according to the similarity of feature vectors within that space [63][36]. Related techniques are then used in image testing to allocate a test feature vector to its associated object cluster.

The deterministic approach developed represents a new industrial application of pattern recognition techniques previously confined to the fields of mathematical statistical analysis and theoretical computer science. To produce a flexible 'third generation' vision system, for the given application the system must have the following characteristics:–

1/ On being presented with a range of items in different orientations, the system must be able to determine automatically the number of object classes represented and the feature characteristics and associated orientation properties of these object classes, to facilitate robot manipulation.

2/ The system must be able to generate automatically a series of decision functions which partition the feature space into the above object classes to facilitate automatic object class identification within the available processing time. It must also be possible to easily extend the decision functions to recognise additional object classes.

3/ For each object class identified the system must be able to reference the associated orientation and pick-up point information and determine the object pick-up point or other important spacial coordinates.

To meet all the above requirements a hyperspace clustering technique was adopted. Techniques in this field come under the definitions of prototype determination and cluster seeking. Cluster seeking is an area, it is conceded by Tou and Gonzalez [36],that is 'very much an experimental art in the sense that the performance of a given algorithm is not only dependant on the type of data being analysed, but is also strongly influenced by the chosen measure of pattern similarity and the method used for identifying clusters in the data'. It is also recognised that these concepts provide the foundations for unsupervised pattern recognition systems. To give a background to the clustering approach there follows a resume of the reason for this choice of approach and a brief description of cluster analysis in general.

A 'case' in the following descriptions is an individual set of features of an item in an image and constitutes a point in the 4 dimensional feature space.

10.1 The Use of a Cluster Analysis Technique.

For this project, one of the most important requirements of the vision system was the need for it to operate in real time. For this application a processing time target of five seconds per complete image was taken. This meant fine tuning the system software for maximum speed. The preprocessing algorithms were coded for maximum efficiency but because of the parallel and repetitive nature of many of these operations, for example the thinning and edge extraction operation, much of this processing would be best implemented on specialized hardware. The recognition stages however, offered the opportunity for considerable processing time savings through the use of well designed and efficient software.

In order to recognise an object from its features, a comparison must be made with a-priori known sets of feature information. In the project task there are usually either two or four hundred separate sets of feature information (cases) for each item to be recognised. This arises from taking sample images in the learning stage at 0.9° intervals when the item is rotated through 180° or 360°. There is also no specific limit on the number of items to be recognised. A way the comparison with a-priori

known sets of feature data might be made is to first normalize the input test features from the feature extraction stage and then calculate the Euclidean distance to each a-priori known case. The nearest feature set would then be taken as matching the object under test.

This approach has two main drawbacks. The first is that it would introduce inaccuracies of identification due to using a 'most-like' criterion rather than an 'actually-exists' criterion for object identification. Figure 10.1 is a two dimensional representation of the feature space. A number of learned cases are shown in the form of blue crosses. These comprise two distinct object classes indicated by the red rectangles. If the above 'most-like' identification method were used the cases marked with yellow crosses –which are nearer to the centres of the object classes than are some cases actually in the object class, would be identified as being members of these object classes. This would be an erroneous classification as it is known from the object learning stage that no objects to be recognised occupy these regions of feature space. Hence in these situations it should be concluded that the cases have resulted from random debris or unresolvable overlapping or interfering shadows.

The use of a fixed distance threshold to reduce this effect also causes problems in the situation of diffuse clusters. This problem is indicated by red crosses in figure 10.1 which show where cases that should actually be classified as cluster members are actually outside the threshold represented by the yellow circles and hence would not be identified as being members of clusters.

The second drawback is that this approach would require many processing operations to calculate the distance to each case, especially if a large number of items was to be recognised. This would introduce a large processing time overhead with a processing time dependent on the number of items to be recognised.

The alternative method used in the project uses cluster analysis techniques to group sets of features with similar properties to reduce the number of case comparisons necessary. These techniques also determine where in the feature space points 'actually exist'. The clustering program runs off-line and provides information on the ranges of features in clusters, the component members of clusters, and pick-up point information about clusters for each item to be recognised. This information is built up into a library of items and used in an image testing program to generate a large identification look-up table in microcomputer memory. A set of test feature data is then scaled and mapped into this table in a simple operation. Identifica-

tion times are hence considerably smaller than would be achieved by the alternative method described. The processing time for this method is also independant of the number of items to be recognised.

The three programs that go to make up the complete vision system; the Learning Program, the Cluster Program, and the Identification Program, are described in the following sections.

10.2 The Learning Stage.

This is the stage at which the a-priori information about items is deduced. As described earlier the mathematical prediction of the behaviour of features was considered impractical due to the complexity of the projection geometry and digitization error functions. Instead a learning-by-teaching approach was adopted which involved presenting the vision system with the items it will need to recognise at a subsequent testing stage, in all their anticipated orientations. Theoretically these orientations should include all an item's stable 3-dimensional resting positions. In the project, for development purposes, these were limited to the rotational variations of upright items, on a flat surface.

The image capture part of the learning stage is fully automated. The item to be identified was placed on the Apple microcomputer controlled rotary table. The program within the Apple controlled the indexing operation. Figure 10.2 is a flow diagram of this program and the actual program is described in appendix AP1.3.1.1. The program allowed the user to specify the rotation parameters which involved indexing the item through up to 360° , depending on the number of axes of symmetry the item had, in 0.9° steps. At each 0.9° rotation step an image was captured and transferred to the DUMC M68020 unit. Figure 10.3 is a flow diagram of the main coordinating program running on this system (appendix AP1.3.2.1.1). The image was read in using the same transfer routine as in the first project (appendix AP1.2.2.4) and preprocessed by making sequential calls to the preprocessing routines described in Chapter 8. The feature extraction routines *-shadw()*, *freri()*, *frearea()*, *paction()*, *ccircle()*, and *dcentre()*, described in the previous chapter then extracted the identfication and pick-up point feature information. Printouts, plots and tables of derived features were produced on a printer and stored on DUMC unit disc by the function *toffile()* (appendix AP1.3.2.1.18) and on DUET-16 disc by the

function *toduet()* (appendix AP1.3.2.1.20). The results of this process are illustrated in section 12.3 of results Chapter 12.

The nature of the chosen features leads to the categorization of items in continuous ranges of angular rotation into the same object classes. This is because they occupy the same region of the chosen four dimensional feature space. An example of this from past experience is of a plate with its major axis between about 10° to the line-of-centres from the item to the light source to the same axis being at 80° to this line. The behaviour of the shadows and the features derived from them, when the item is rotated cannot be relied upon to give continuous mathematical functions except for the most simple of items. Refer to figures 10.4 to 10.11 for plots of the features of a bowl and cup as functions of the item's angular rotation. In the bowl plots the bowl starts with its long axis at right angle to the incident light direction and the in the cup plots the cup starts with the axis through the centre of the cup and handle at right angles to the incident light direction.

The figures illustrate the erratic peaks and troughs in the functions and the profiles indicate the rates of change of features at different angles of object rotation. It can be seen that sometimes these changes are very abrupt and have resulted in near spike-like behaviour. As anticipated this effect is most pronounced when an object edge is becoming aligned with with the incident light direction – refer to section 9.3.

(A detailed investigation and explanation of the shadow behaviour in these plots is given in results section 12.3.)

The irregular behaviour of these measured features makes it clear that the identification features could not be modelled or predicted without difficulty. Although it is possible for the experienced system programmer to make the object class divisions intuitively, the use of cluster analysis techniques meant that a versatile vision system could be produced requiring a minimum of skilled supervision.

In the project the learning stage item feature data was made available to a separate clustering program which contained suitable clustering algorithms to perform the following functions:–

1/ They divided up the feature ranges so that they corresponded to previously undefined groups of cases.

2/ They took into account the ranges of relative pick-up point coordinates of objects in the clusters so that the cluster divisions gave the required accuracy in robot pick-up point location.

3/ They provided cluster information in a form suitable for use by identification and orientation routines in a subsequent image testing phase.

The next section briefly describes the general principles of cluster analysis. This is followed by a description of the particular method adopted.

10.3 Cluster analysis.

Clustering can be defined as the classification of variables or cases specified by a number of variables, into groups with similar properties. When processing visual images these variables correspond to a series of image features. From experience it is assumed that by grouping similar ranges of features, derived from sample test images in a learning phase, a small number of clusters corresponding to object classes will emerge.

Essentially all clustering approaches are concerned with the minimization of some inter-cluster criterion and the maximization of an intra-cluster criterion. According to Hartigan (1975)[64] in his definitive book on cluster analysis there are six main approaches to forming clusters:–

1/ Sorting. – use an important variable to give an initial class division and then make subsequent divisions using less important variable in turn.

2/ Switching. – make an initial partition and then move objects around according to some criterion to get optimal clusters.

3/ Joining. – start with clusters containing a single case and join the the closest pair, continuing until you get a single cluster containing all objects.

4/ Splitting. – partition objects into several clusters and then partition further.

5/ Adding. – start with a cluster structure and add each object in turn to

the nearest cluster.

6/ Searching. – in some cases particular mathematical criteria can rule out clusters. In this case the remaining cluster possibilities can be searched for an optimum classification.

In order to decide which of the techniques listed was suitable for the industrial imaging application the following factors were considered:–

1/ Does the method generate the clusters required to distinguish the anticipated tray items and their orientations ?

2/ Does the algorithm provide classification into the relevant object classes so that sufficiently accurate pick-up point calculating algorithms can be applied to the objects identified?

3/ The processing involved in determining the clusters is 'off-line' which means that computing time involved at this stage is not a major consideration. The method must however, produce clusters that can be discriminated at high speed in the testing stage.

4/ The algorithms must be able to operate within the limitations of the hardware available; for example memory storage.

5/ Ideally the method must permit easy augmenting of the learnt object classes to include new objects.

To investigate which of the possible clustering methods was most suitable it was decided to apply some standard mainframe statistical analysis programs to sample image feature data. The particular packages chosen were within BMDP (Biomedical Data Processing) [65] – a suite of programs on the Newcastle MTS system which offers a number of different multivariate and cluster analysis programs.

The cluster analysis options fell within two main categories, single-linkage and K-mean cluster analysis. Within these programs there were options to specify the inter-case/cluster distance measure and standardization functions between data. These allow features of different magnitudes and ranges to be normalized and to be given a weight corresponding to their relative importance. The standardization method used in both of these programs and in the programs written to run on the dedicated M68020 system was to divide each of the equally weighted recognition

features by its standard deviation. This meant that data which varied more was effectively scaled down in significance.

The image feature data to be processed was generated on the vision system, stored on a DUET-16 microcomputer disc and from this system transferred to the mainframe.

10.3.1 Single linkage clustering.

This technique falls under the category definition of a 'joining' algorithm. Each case, specified by four standardized coordinates in a 4-D hyperspace, is initially given its own cluster. After this allocation the two nearest clusters are combined to give a single cluster. This process continues until all clusters are contained within one large cluster.

The options possible when running this program include the way of defining the proximity of one cluster to another. This depends on whether a cluster is still considered a collection of cases (to any one of which a distance measure can be made), or whether a cluster is defined by a centroid point and distance measures between clusters made with respect to these.

For near-homogeneously distributed points this choice may well have a significant effect on the order in which points are assimilated into larger clusters. Also if 'case-to-case' distances are used the single linkage method can produce strung out clusters with far off points and clusters connected by a string of nearby points. This is only satisfactory if the clusters are anticipated to have sausage shapes. The 'case-to-case' option also depends strongly on the measurements of small intercase distances which can lead to unstable clusters if they are not accurately measured. Overall 'average joining' can be considered to produce more stable compact features. The sequential linking of clusters can readily be represented as a diagramatic tree structure which offers an easy visual interpretation of the cluster structures. With either approach however there is the problem in deciding on how real any of the clusters represented are. This involves selecting just a few significant clusters from the many produced and ensuring that they are not, for instance, a result of multivariate normal distributions.

Suggested ways of achieving the above selection are described in Hartigan [64]. They have included the 'moat diameter' approach [66] as a measure of inter-

cluster isolation, and minimum spanning trees which if chosen at a mode point the inter-point gaps steadily increase as objects are successively added from regions where the point density distribution is decreasing. The measure of cluster size then comes from an investigation of the monotonicity of the gap sequence (consider section 10.4.2./1).

Other approaches to finding clusters from the single linkage tree involve the cutting of links of length greater than G (a chosen threshold), or take the singly-linked large cluster and remove succesively the largest links.

10.3.2 K-mean clustering.

In this approach one starts with an initial partition of the feature hyperspace into a number (K) of clusters and then one proceeds to search through the set of partitions moving a case from a given partition to that partition in its neighbourhood for which some error function is minimized. The search stops when the error function e[P(M,K)] is not reduced by movement of its neighbourhood. The partition is in this case locally optimum.

For given standardized and weighted variables and a given inter-case/cluster measure the options with the K-mean algorithm are:–

1/ The number of starting clusters.

2/ The case movement rule.

3/ The cluster updating rule.

The criteria for evaluating the performance of the algorithm are in the expected time of calculation (which is of minor importance in the project case) and the difference between the local and global optima. It is also desirable that the clusters be independent of the order of input of cases. To achieve this it may be necessary to re-order cases initially.

Starting options.

1/ Choose starting clusters at random.

2/ Choose a single variable and divide it into K intervals of equal length. (This 'variable' might be the average of all variables or the weighted combination that maximizes variance).

3/ Let the starting clusters for K be the final clusters for K-1 with that case furthest from its 'cluster mean' split off to form a new cluster.

Movement options.

1/ Run through the cases in order assigning each case according to the 'cluster mean' it is closest to.

2/ Find the case whose re-assignment most decreases the 'within cluster' sum-of-squares and re-assign it.

3/ For each cluster, begin with zero cases and assign every case to the cluster at each step, finding the case whose assignment to the cluster most decreases (or least increases) the 'within cluster' sum-of-squares. Then take the cluster to consist of those cases at the step at which the criterion is a minimum. This makes it possible to move from a partition which is locally optimal under the movement of single cases.

Updating options.

1/ Recompute the cluster means after no further assignment of cases decreases the assignment error.

2/ Recompute the mean after each assignment.

The computational steps in the mainframe program are given in Appendix 2.

10.4 Mainframe Results.

The two algorithms were applied to the actual bowl feature data in figure 10.12. The form of this illustrative feature data file is described in detail in section 12.3. The file contains data for images taken at 1.8 degree rotation steps to reduce the number of data sets in the printouts. In practice for cluster determination in the learning stage 0.9 degree steps were used.

Printouts of the results from the two approaches can be seen in figures 10.13 and 10.14. The K-mean clustering results in figure 10.14 are included to allow both a comparison between the single linkage results and to evaluate the performance of the K-mean program written for the dedicated system.

The option of single linkage joining for the single linkage clustering was used

as the clusters were anticipated to have extended shapes. For the K-mean clustering start option 3 was chosen, and all points were assumed to belong to a single cluster initially. This structure was broken down using movement option 1 and updating option 1 until the specified number of clusters was reached.

10.4.1 Single linkage results.

In the single linkage program printout (figure 10.13) the output stages are:-

1/ A listing showing the control and data parameters used in the program.

2/ This is a tree diagram (dendogram) of the clusters. The labels of the cases are printed to the left of the diagram. Cases are sorted to permit a tree to be drawn. Across the top of the tree the amalgamation distance increases from left to right. Each small line at 60° from left to right (/) in the tree corresponds to a cluster formed in the amalgamation process. The clusters formed by grouping cases can be seen by following the broken sloping lines made by these sloping line sections.

3/ The initial distances between each pair of cases are ordered according to the tree diagram and printed in a shaded form as indicated.

10.4.2 K-mean clustering results.

In the K-mean program printout (figure 10.14) the output stages are:-

1/ A listing showing the control and data parameters used in the program.

2/ Two histograms displaying the distance from cluster centre to each case, a) for cases in the cluster, b) for cases not in the cluster.

3/ The cases in the clusters are listed with their weight and distance from the centre of the cluster.

4/ The three univariate statistics are computed from the standardized data for the cases in the cluster.

5/ Cluster means and within cluster standard deviations are computed from data in the original scale.

6/ Analysis of variance of each variable, comparing the between-cluster mean square to the within-cluster mean square. F-ratios can be used to

describe differences between variables.

7/ Cluster profile plots – for each cluster the mean is marked by a numeral and a ± standard deviation by a dashed line. The vertical line connecting the asterisks above and below each cluster show the position of the grand mean.

8/ Pooled within-cluster covariances and correlation matrices.

If a sequential maximum link length cutting method is used in the single linkage approach to produce clusters as indicated by cluster numbers C1 to C8 on figure 10.13, and the resulting cluster case divisions compared with the equivalent K-mean cluster numbers in figure 10.14 the comparison suggests that for the given data the divisions are, essentially, independent of which of the clustering techniques is used. As more step-by-step control over the clustering process was provided by the K-means algorithm, with the ability to take into account other factors during the clustering, and to split existing clusters easily this approach was chosen for the project.

10.5 The Cluster Approach Chosen.

The main external factor to be taken into account during the production of clusters for the project was the accuracy required in the pick-up point coordinates.

The pick-up point coordinates are defined by three radius arcs from the three unique, conditionally defined points on the inner edge of the shadow outline (figure 9.1). The intersection of these arcs gives the point where the object is to be picked up. Once the cases in a cluster have been determined the radius distances to the three points are obtained by bisecting the mimimum and maximum distance to the pick-up point for each radius. Hence the additional clustering criteria for objects in a cluster was that the range of variation of any one of these radii could not be more than twice the required accuracy in pick-up point determination.

For a simple item, for example a cube, the variation in these radii may be small despite large ranges in the identification features. In this case cluster divisions based only on pick-up features would lead to unnecessarily large clusters, containing cases with widely different characteristics, because of the rectangular nature of feature clusters in the recognition stage. Figure 10.15 is a schematic two

dimensional representation of this situation. In figure 10.15a there is a hypothetical representation of the variation of a cube's shadow area with angle of rotation. Figure 10.15b illustrates schematically the anticipated small variation of the 'distance-to-midpoint' feature over the same rotation angle. It can be seen in figure 10.15c that if cluster distinctions were based only on the basis of the variation of this pick-up point feature the result would be the production of a small number of unwieldy clusters with large ranges of area or other identification features. These would occupy large areas of feature space on account of the feature ranges for the cluster being taken from the maxima and minima over all cases in that cluster. Hence for this reason it was necessary to first make a division into clusters using a normal inter-cluster distance measure before applying the distance clustering criteria.

10.5.1 Cluster program software.

Figure 10.16 is flow diagram of the resulting K-mean algorithm implemented on the dedicated system and figure 10.17 is a printout of the VDU screen output for a program run on the same data as was used for the above mainframe program runs. Figure 10.17 is included so that the results of the dedicated K-cluster program can be compared with those of the mainframe program.

The main coordinating function first calls *cache()*(appendix AP1.4.1.1) and *initdu()* (appendix AP1.4.1.2) to initialize the hardware. After a greeting presentation has been produced on the VDU screen by the function *greet()* (appendix AP1.3.2.2.2) the routine *cluscrit()* (appendix AP1.3.2.2.3) is called which allows the user to input the external clustering criteria for the clustering process. *fromfile()* (appendix AP1.3.2.2.4) reads in the specified feature data file by means of function calls to the disk system interface subroutines and the input feature data is standardized in the function *standardize2()* (appendix AP1.3.2.2.5), as in the mainframe program by dividing each of the variables by its standard deviation. The program loops checking the first clustering criteria in the function *crit1()* (appendix AP1.3.2.2.6) which is a test of whether the required initial number of clusters has been reached. Within the loop the standard deviation and mean of each cluster generated are calculated in the function *meansdnp2()* (appendix AP1.3.2.2.7). *varmaxt1()* (appendix AP1.3.2.2.8) determines the cluster with the feature with the maximum variance and *clusplit()* (appendix AP1.3.2.2.9) splits this cluster. After the first clustering criteria has been met the routine *move2()* (appendix AP1.3.2.2.11) moves cases into the cluster whose

centre is nearest.

The second program loop is now entered in which the second clustering criteria is tested by *crit2()* (appendix AP1.3.2.2.12). This determines the ranges of pick-up point radius for each cluster and tests whether the range for any cluster is greater than an acceptable limit. The loop is similar to the previous one except the function *varmaxt2()* (appendix AP1.3.2.2.13) takes the cluster with the largest range of pick-up feature and finds the identification feature with the maximum variance. The function *clusplit()* then splits this cluster. When the second criterion is met the routine *move2()* is called again. As this routine can affect the composition of clusters it can also change the ranges of pick-up point features in the clusters. For this reason the pick-up point feature loop is repeated and empirically it is found that there is seldom any further movement of cases at this stage. The routine *detinfo()* (appendix AP1.3.2.2.14) determines the cluster and pick-up point information to be stored on disc so that the clusters can be reconstructed in the image testing program. The routine *storefile()* (appendix AP1.3.2.2.15) then generates a cluster data library file on the DUMC M68020 unit disc operating system to store this information.

The cluster information is in the form of the maxima and minima of each feature for each cluster. Theoretically the correct form to store this information would allow the complete 4-D convex hull of the cluster to be reconstructed. The chosen storage method simplifies and rectangularizes the clusters, considerably reducing the amount of information needed to be stored to reconstruct the cluster. This rectangularization procedure is a potential source overlap of clusters due to the overlap of the corners of clusters determined by spherical distance measures. It is expected that clusters that would overlap in this way would probably come from the same item with similar values for pick-up point radii. In practice it was found that a number of cluster collisions did occur but these were mainly due to quantization effects in the look-up table (refer to section 12.5).

It should be noted in this software that the second cluster splitting operation only uses the pick-up point features to determine which clusters are to be split. It plays no part as a clustering variable. If these features were to be treated as clustering variables in the same way as the identification features, cluster divisions would be made which depended on the measurement of information not available in the image testing phase when the pick-up points would not be marked. This could well introduce ambiguity and error in identification into the recognition system.

10.5.2 Cluster program performance.

The printout of the program run in figure 10.17 shows the results the clustering program produces from the given set of bowl data. It can be seen at point X that the sets of feature data have initially been partitioned into 8 clusters. These are indicated by coloured line sections on figures 10.4 and 10.5. If these clusters are compared with those clusters produced from the same data by the K-means mainframe program in figure 10.14, for example if 1, (2 and 7), and 3, are compared with mainframe clusters 4,5, and 2 respectively, it can be seen that there is a strong correspondence between the results from the two clustering programs.

In the dedicated program the subsequent operations have split these clusters further to provide the required pick-up point accuracy. It can be seen that, compared with clustering operations based on the original criteria, the cluster distinctions are slightly different from those that would be produced by the normal splitting operation alone. This is exemplified by figures 10.17 and 10.18 which contain the resulting clusters for the bowl. Figure 10.18 has an initial cluster requirement of 14 clusters with effectively no pick-up point accuracy requirement. Figure 10.17 has an initial requirement of 8 clusters with a required accuracy of pick-up point location of 7mm. to also give fourteen clusters. The difference is demonstrated by considering final cluster 3 in figure 10.18. It contains only five cases, 45, 46, 54, 55, and 56, with a range of pick-up point to maxpoint feature from 33 to 46 pixels. In the run where a 7mm (12 pixel) accuracy has been specified (figure 10.17) cluster 3 has been split to form new cluster three with cases 45 and 46, and cluster 10 with cases 54, 55, and 56. The maximum range of any pick-up point feature in either of these clusters is 4 pixels or 2.5mm.

10.6 The Object Inspection Program.

After building up a library of cluster data files, one for each item to be recognised, the system is ready to sample test images. The Apple program in appendix AP1.3.1.2 controls the Apple to provide raw binary images of the inspection table on the high speed parallel link to the M68020 unit. Figure 10.19 is a flow diagram of the Apple image capture and transmit program. Figure 10.20 is a flow diagram of the complete testing sequence on the DUMC M68020 system. The *main()* coordinating function (appendix AP1.3.2.3.1) acts in the same way as the coordinating function in the learning program, both to declare and initialize the external variables and then to call the functions in sequence.

First in a parameter setting function *changep()* (appendix AP1.3.2.3.2) the user sets program options including printout and other input/output options, including the input image source, either from the Apple or from internal disc storage. A look-up table coordinating function *lookup()* (appendix AP1.3.2.3.3) is then called which calls *rdbkdata()* (appendix AP1.3.2.3.4) to read in specified library cluster data files of items to be recognised and *map()* (appendix AP1.3.2.3.6) the look-up table creating function. In addition a number of optional utility functions can be called at this stage to present the cluster information and test the look-up table.

The function *map()* takes the maximum and minimum of each feature over all item clusters to be recognised and calculates their ranges. A scaling factor is generated for each feature so that a measured feature for an item to be recognised is scaled to be between 0 and 15 inclusive for the minmid and midmax features, and 0 and 31 inclusive for the minmax and compaction features. A look-up table is then generated by mapping the cluster feature ranges into a 250K×2 byte array (4 features (2×5 bits + 2×4 bits) of address) of all the known clusters into the microcomputer memory. This mapping ensures that the features are orthogonal. Each memory location accessed has a value placed in it signifying the item file and cluster number for that item file. If two different clusters map to the same location due to a cluster collision as described previously, a collision index pointer is placed in the location which points to a list of the colliding object classes for that location.

Control of the program after the lookup table generation stage returns to the main function which enters the image testing sequence. Images are read in from the specified source and the necessary processing operations performed on these to extract the features of the shadow outlines as in the learning program (appendix AP1.3.2.3.8). Next an area correction routine *arcorrect()* (appendix AP1.3.2.3.9) is called to correct for the geometric distortion of shadow areas due to the overhead position of the camera (see Chapter 7.1). The function *identify()* (appendix AP1.3.2.3.10) is then called which takes the features of significant image outlines and scales them by the above scaling factor. An address into the look-up table is then generated by a hash type function and the object identified directly by the look-up table entry at the accessed address. The entry also indexes into an array containing further identification and orientation information which allows the calculation of the identified item's relative and absolute pick-up point by a method of intersecting arcs. This operation is described in the following chapter on pick-up

point determination.

This look-up table approach has an identification-time independant of the number of items to be identified for a given screen resolution. This is in contrast to the inter-feature set distance measure method described previously and is unlike a decision tree approach whose complexity and required computing time increases dramatically with the number of clusters to be identified. In addition the mapping of known objects into the look-up table means that the system is able to make a definite negative identification when an object does not match a known object instead of producing a 'best match'. Under certain circumstances in a more general industrial application this may be equally as important as a positive identification, for example where guiding a robot to attempt to pick up an unforseen object may damage the robot or object, or where due to an incorrect identification a robot may pick up an object insecurely and hence represent a hazard to nearby human operators.

The chosen approach capitalizes on the decreasing cost of micrcomputer static RAM now standing at about £5 for 32 kilobytes. If a known variety of objects were to be identified, at the cost of a temporary loss in system flexibility, the approach would readily allow for the substitution of the dynamic-random-access memory (RAM) elements by eraseable programmable read-only-memory (EPROM) elements in the look-up table. The cost of these EPROMs has fallen as low as £5 per 512 kilobit chip. With essentially no restrictions on M68020 microprocessor address space this system has the potential for the production of systems with extremely large memories –for example of greater than 40 megabytes for less than £3000, which would be capable of recognising a large range of varied objects, with a recognition time independant of the number of objects to be recognised. An extention of this increased look-up table memory size would be to use one of the new, comparitively extremely low-priced Winchester hard disc drives in the place of the RAM look-up table. Whilst preserving the system flexibility with modifiable memory elements this would allow 100 Mbyte memories to be used at a token cost of an extra £1000. The increase in on-line processing time would be minimal if only a few accesses were to be made for identification purposes as Winchester access times are of the order of milliseconds. The addition of such a unit to the existing system offers the potential to produce an extremely powerful system using more features with increased resolution for more reliable object identification and orientation.

CHAPTER 11

ITEM PICK-UP POINT DETERMINATION

A pick-up point as defined for the project task is the point on an object which it would be gripped by if it were to be picked up from above. In the case of the particular project items it is the point where a vacuum chuck would be guided to in order to pick-up the items.

This chapter describes the way these points are calculated for identified items. These calculations are at the end of the image processing sequence and are only applied to recognised objects. As the routines are only likely to be called upon to find item pick-up points a maximum of ten times the time penalty for the slow floating point operations necessary to give the required geometrical accuracy in pick-up point calculation is small.

The software has been designed so that the tray items can be picked up at a single point. This would be the case for example if one was picking up the items using a vacuum chuck from above. The software for pick-up point learning and calculation could readily be modified however, to cater for the use of multiple contact grippers using the same basic routines and a variety of differentiable pick-up point markers.

The way the pick-up point features are calculated in the learning stage is described in Chapter 9.4.3. These features are essentially radius arc measurements from the centre of a pick-up point marking circle placed on the items at an intended pick-up point, to the three conditionally defined points on the inside edge of the shadow, the minpoint, the midpoint, and the maxpoint (figure 9.1). The clustering algorithm described in Chapter 10.5.1 takes the ranges of these measurements into account when producing clusters, to ensure that no cluster contains a group of cases for which the range in the maximum to minimum pick-up radius is greater than twice the required accuracy in pick-up point location. The bisector of this range for each feature is taken as the feature radius for all cases in that cluster. Although the use of the median would offer greater accuracy in most cases this would not guarentee the required accuracy for all cases in a given cluster.

11.1 The Pick-up Point Software.

The software comprises two main functions called at the end of the test image processing sequence. After the routine *identify()* has been called and objects identified the library item file and cluster number are passed to the routine *circept()* (appendix AP1.3.2.3.11). For each identified object the routine accesses the associated pick-up point radius and conditional point coordinate information. It then calculates the coordinates of the three arc intersection points. Figure 11.1 is a schematic diagram illustrating the pick-up point finding geometry. The figures in results Chapter 12.4, figures 12.41 to 12.50 are plots of the actual arcs calculated in determining the pick-up points of a bowl and a cup. The calculation involves the solution of the simultaneous equations for the circles with centres at the conditional points.

Referring to figure 11.1

The equations for the circles are:–

$$R_1^2 = (x - X_1)^2 + (y - Y_1)^2$$

$$R_2^2 = (x - X_2)^2 + (y - Y_2)^2$$

$$R_3^2 = (x - X_3)^2 + (y - Y_3)^2$$

Solving the first two equations simultaneously for x gives:–

$$x^2(1 + B^2) - x(2X_1 + 2BY_1 + 2BA) + X_1^2 + Y_1^2 + A^2 - 2Y_1A - R_1^2 = 0$$

Where:

$$A = \frac{(R_1 - R_2)(R_1 + R_2) + (X_1 + X_2)(X_1 - X_2) + (Y_1 + Y_2)(Y_1 - Y_2)}{2(Y_1 - Y_2)}$$

And,

$$B = \frac{(X_2 - X_1)}{(Y_1 - Y_2)}$$

In the solution of the quadratic equation in x above the case of $b^2 < 4ac$ in the standard form of the quadratic $ax^2 + bx + c = 0$, gives an unsolvable negative square root solution. In the routine *circept()* this is considered to be due to inaccuracy in measurcment of the positions of the endpoints and in this case the single root solution to $b^2 = 4ac$ is taken corresponding to a single point of contact for the two circles.

If the two roots to the equation do exist, that is $b^2 > 4ac$, then the two intersection point coordinates are calculated, one for the positive and the other for the negative square root solution. This can hence lead to a maximum of six possible arc intersection points.

The next function to be called *-cockhat()* (appendix AP1.3.2.3.12) takes the leftmost midpoint arc intersect with each of the two other circles as the true intersection coordinate solutions. The intersection point of the other two circles that is nearest to the midpoint of the line joining these two points is taken as the third true intersection point. In the absence of any reliability factor being considered for any of these points according for instance to the variance of the pick-up point feature for that cluster these points must be considered to have equal weight. The three points are considered to define a triangle and in an operation analagous to the cock-hat operation in navigation, the position of the pick-up point is found by determining the centre-of-gravity of the triangle. From geometry this is known to be one third of the way along a line from a vertex to the bisecting point of the opposite side (figure 11.1).

It is known that if there is an equal probabity of the arc direction being in error in the clockwise direction as in the anticlockwise direction then the probability of the true point actually occuring in this area is $(\frac{1}{2})^3$, or $\frac{1}{8}$. As this area represents the greatest probability per unit area that the centre occurs here then this is assumed to be the best point one can achieve.

The routine goes on to calculate the coordinates of the points of a cross with its centre at the pick-up point for each identified object. The optional outputting routine *shipcen()* (appendix AP1.4.1.6) can then be called to output the x and y coordinates of these crosses to the screen plotting routine on the DUET-16 microcomputer. If this is done in conjunction with calling the optional line section plotting routine *shipcornpw()* (AP1.4.1.5) which outputs the tracked line section plots of the image, the resulting plot gives an indication of the accuracy of pick-up point location. This is made more obvious when the learning stage pick-up point markers are left on the items in the testing stage (Refer to figure 12.36 to figure 12.50 in results Chapter 12).

11.1 Potential Pick-up Point Error.

The possibility of a pick-up point error greater than the specified accuracy in the clustering program for a correctly identified object could be anticipated to arise when, if one of the special points on the inside edge of the shadow is slightly in error, (for example the midpoint on vertical section lines can often 'wander' to a value different from in the learning program), and push the triangle out of shape to the extent that it is out of the accuracy range for the other two pick-up radii. This effect of the special points being in error should however, be self correcting to a large extent. This is due to the change this will cause to the measured inside edge alternative features and hence to the object class identification. The change will cause the object to be associated with a different learnt object class with the appropriate pick-up point radii.

The performance of the system in terms of how accurately the pick-up point is calculated can be seen in Chapter 12 section 12.4.

CHAPTER 12

SECOND PROJECT RESULTS AND CONCLUSIONS

This chapter illustrates the results obtained at different stages of program development. The vision system software comprises three main programs – the learning program, the clustering program and the identification program. The first section of the results is concerned with establishing the methodologies and parameters which have a bearing on all the programs. In the following two sections the results produced by the individual learning and recognition programs is presented. Following this there is a discussion of the results and a conclusions section in which the overall performance of the system is considered.

The performance of the clustering program is fully investigated in Chapter 10.

12.1 Presentation of the Results.

The printouts of program runs were produced on an Epson FX100 dot-matrix printer by redirecting the serial print output stream from the terminal to the printer. The original binary and thinned binary image printouts were also produced on the Epson printer. The routine *pripwb()* (AP1.4.1.3) controls the printer to operate in bit-image mode. In these printouts one black dot corresponds to a single positive pixel in the image.

The routine *printable()* (AP1.4.1.4) produces a results table on the printer in which the features for each outline above the length threshold is tabulated. The first nine columns contain the i, j, and point identifier number (the number of the particular point along the tracked outline) for the minpoint, midpoint and maxpoint respectively, along the line. (Note that the i-coordinate increases from left to right over the range 0–279 and the j-coordinate from top to bottom in a range from 0–191). The next four columns show the chain-coded features – the number of points on the outline, its perimeter length, the enclosed shadow area, and the heuristic compaction factor (perimeter^2/area). The alternative features follow in the remaining columns: the minpoint-maxpoint distance, the minpoint-midpoint distance and the midpoint-

maxpoint distance. The 'total' feature is the sum of the last two of these features. Some of these tables also contain a number of other interpoint distances.

When referring to a features table for a given image, to find boundaries of significant objects it is usually only necessary to scan the horizontal rows for complete sets of features. These are rows for which none of the features is zero. One can then refer back to the minpoint i and j coordinates and match these with the approximate coordinates of the minpoint of the object of interest in the image.

The coloured line section plots were produced on a Hewlett-Packard graphics plotter. The routine *shipcornpw()* (AP1.4.1.5) outputs those points which are a change in tracking direction in the chain-coding stage, of line sections above the chosen threshold length, to a DUET-16 microcomputer which is running the *'Read-in-and-store'* program (AP1.4.2.1). The data disc is then transferred to a DUET connected to a plotter at a different site and the *'Disc-read-and-plot'* program run. This routine produces a plot between these points. The changes in line colour serve to indicate distinctly where nearby line sections start and finish.

12.2 Investigating General Progam Parameters.

12.2.1 The effect of different thinning algorithms.

This section shows the effect of applying different edge extraction routines to the original image. By considering the printouts and plots of the resulting edge extracted image the section illustrates the factors affecting the eventual choice of a working algorithm.

The results are produced by applying the three developed alternative thinning/ edge-extraction algorithms to images of items in a range of orientations. The first algorithm produces an edge around the positive pixels outside the dark shadow region, the second produces an edge within the dark region of the shadow. The third algorithm is a modified version of the second. In it additional criteria are applied to each individual interior shadow point before it is switched on in the resultant image (refer to Chapter 8.2).

Figures 12.1 and 12.2 are dot matrix, one point per pixel, printouts of test images containing a single cup, bowl, and plate. These original images are limited to having the items in them not in close proximity to, or in contact with, the other

items. In the first test image, figure 12.1, the items have a skew axis orientation and in figure 12.2 the items are almost, but not quite aligned with an axis near to the line from the item to the light source. This results in the shadows having narrow tail sections. The resulting images generated by applying the three thinning algorithms to these original images are given in dot-matrix form in figures 12.1a, 12.1b, 12.1c and 12.2a, 12.2b, 12.2c respectively, with their associated feature tables. The line section plot forms of the resulting tracked outlines are in figures 12.1a1, 12.1b1, 12.1c1, and 12.2a1, 12.2b1, 11.2c1.

In the case of the second thinning algorithm, the chain-coded outline produced from the thinned image outline is considerably more fragmented than that produced by either of the two other algorithms. This can be seen by comparing figure 12.1b1 with 12.1a1 or 12.1c1. The application of the algorithm around individual shadows has in every case resulted in an outline of two or more separate boundary line sections, usually with one section representing the rear edge of the shadow and another the front. This is as a result of the production of single-point-width line sections as anticipated in chapter 8.2, which act as 'dead-ends' to the tracking algorithm subsequently applied. In the case of these split boundaries, unless an additional more complex resegmentation procedure is applied to the boundary, to recombine outlines, the derivation and subsequent use of the standard chain-coded features, such as area, perimeter, and compaction factor is precluded. In this split boundary case the reliability and confidence in the subsequent outline identification is reduced, with only length based features available to make the identification. In such instances there is usually a small compensation. When an outline is split there are usually two separate outlines, one for the front and one for the rear edge of the shadow. As it is possible to perform a test to determine whether a line is a rear edge or not, the reliability of identification of an item is potentially increased by having two lines providing features to identify a given item.

It can be seen that the outlines produced by the first and third algorithms, in dot matrix prints, figures 12.1a and 12.1c, result in similar tracked outline sections in the line section plots in figures 12.1a1 and 12.1c1, and that for the most part the tracked sections are virtually equivalent. The closed shadow sections retain their continuous outlines in the final tracked image. Differences do occur when the algorithms are applied to images which contain narrow shadow tails. For example, compare figures 12.2a and 12.2c. and their line section plot forms figures 12.2a1

and 12.2c1. In these cases the first algorithm produces more clearly defined outlines on these narrow sections. The image detail of these thin trailing sections of the shadows is retained. In the particular problem however, it is questionable whether this retention of detail is necessary as in some cases it down-grades the edges of shadows and complicates, rather than assists, item identification.

If the first and third thinning algorithms are applied to the same test image and the chosen features compared the line based features are seen to be almost identical in size. Compare figures 12.1a and 12.1c, and 12.2a and 12.2c. There is however a small difference in the perimeter size and, as anticipated, a considerably greater difference, of the order of 25%, in the area magnitudes. This suggests that the two thinning algorithms cannot satisfactorily be used interchangeably in the identification program.

In another test of the performance of the thinning algorithms the first and third algorithms were applied to an image more representative of the proposed long term application. This image in figure 12.3 has items randomly arranged and in close proximity to other items. Figures 12.3a1 and 12.3c1 show plots of the resulting thinned and tracked outlines. It is clear that with the given resolution of the image the first algorithm produces outlines which run into each other – for instance the outline of the cup on the left with the plate below it, and the outline of the bowl on the right with the cup below it. This clearly makes the task of identifying an individual object from such an outline considerably more involved, requiring the use of further, more complex, segmentation procedures. In the case of the third algorithm the shadow outlines have been retained as distinct, separate entities. In the case of the plate where the shadow is extremely narrow the third algorithm has had the effect of further narrowing the shadow until the tracking algorithm has crossed from the rear edge of the plate shadow to the front edge and back again. This inevitably confuses the subsequent feature derivation and hence item identification.

In this situation, where the items are inclose proximity to other items on the tray, the third algorithm would appear to give more easily interpretable and reliable results for the given image resolution. Despite this long term project objective advantage, because of the marginally better performance of the first algorithm on the restricted project problem of separate items, in terms of preserving image detail, this algorithm was chosen for use in the subsequent program development.

12.2.2 The effect of changing line threshold length.

This section of results illustrates the effect of varying the 'number of line points' threshold below which a line section in the edge extracted image is ignored for subsequent feature processing purposes.

The printouts in figures 12.4a, 12.4b and 12.4c, show the tabulated feature printouts of typical program runs on the test image in figure 12.1 with the line thresholds set at 0, 12, and 50 points respectively, having used the first of the thinning algorithms from the section above.

The feature table generated for a boundary threshold of 0 in which all line sections are plotted, shows the large majority (36 out of 59) of line sections fall below 5 points in length. The table indicates the feature processing as applied to every line. For very short line sections it can be seen that the attempt to derive features has been unsuccessful: a large amount of unnecessary and time consuming feature deduction has been attempted on many insignificantly small line segments, often only a few points in length. For object identification purposes there is no image information lost by ignoring such short line sections, when no subsequent resegmentation procedure is to be applied. This tactic of ignoring short line sections is in accordance with the gesthalt approach proposed by Pun and Coulon in their work on visual prothesis [67].

With the 12 point minimum length (figure 12.4b) it can be seen that the majority of small line sections are removed but that some unnecessary processing is still being applied to insignificant line sections.

The design of the thinning algorithm means that the thinned shadow outlines of significant tray items produced from the initial clean image, give continuous longer line sections at the tracking stage. In the third case with the line point length threshold set at 50 it can be seen in figure 12.4c that the significant lines are still processed, whilst avoiding the unnecessary processing of the shorter line sections. The number of lines being processed in this case drops from 59 to 16. It is estimated that the shortest line to produce an identification feature for the items to be recognised is of the inside edge of the shadow of a cup. In this case the lower limit line length was estimated at 65 points. This is a little longer than the chosen threshold, but the line-length distribution shows that no significant reduction in the number of lines processed would have resulted from using this value and in

subsequent processing stages the threshold value of 50 points was used.

12.3 The Learning Program.

Refer to figures 10.2 and 10.3 for flow diagrams of the Apple and DUMC unit programs. Figure 12.5 is a printout of the DUMC unit screen output of this program. By comparing these figures it can be seen where each particular function is called.

In the learning program the standard unit of angular rotation specified by the user in the Apple program is 0.9° . In figure 12.6 there is a superimposed plot of four bowl outlines at two step intervals, that is 1.8° , between 0° and 7.2° . The closeness of each successive overlaid plot indicates how small an angle of 0.9° represents.

In figures 12.7(a and b) to 12.28(a and b) there is a sample series of printouts and plots for bowls and cups used in establishing the cluster boundaries for these items. For the bowls the images shown are of items at 18° rotation intervals with the item in orientation from 0° to 180° . An overlay of four such bowl files at 45° rotation intervals can be seen in figure 12.27. For the cups the angle of rotation interval is 36° and the range in orientation from 0° to 360° . An overlay of five cup files at 72° rotation intervals can be seen in figure 12.28. For both of these images the colour plotting order was blue, red, green, and then yellow. For each of the other images there is a printout of the thinned image, a tabulated printout of the identification features, and a plot of the tracked outlines.

It is possible to see from these printouts and plots how shadows change under rotation of the item causing them and that this behaviour is not always that which would be expected by an untrained observer. In the bowl figures the bowl starts with its major axis at right angles to the incident light direction. As it turns the top short edge makes an increasing contribution to the shadow. There is then a fairly continuous stable shadow contribution by both the long and short edge until the long edge gradually dissappears to give a shadow cast only by the short edge. An equivalent process is repeated as the bowl rotates between 90° and 180° .

In the cup figures the handle significantly effects the shape of the shadow. As the cup rotates the illuminated top of the handle disturbs the dark shadow and causes it to be split into two sections. The shadow comes together again as the

rotation angle approaches 180° . Above this angle the handle contribution declines and its shadow becomes separated from the main cup shadow. Over the remaining angles, up to about 330° , the handle makes no contribution and the shadow shape is comparitively constant. Above this angle, up to 360° , the handle's shadow again becomes relinked to the main shadow.

The values of all the features at 0.9° intervals over the required rotation range are stored in a feature-data file which has the form of figure 10.12. This is an example bowl feature data file with a 1.8° rotation step. The columns going across the file represent the following features: 1/ The angle index (units of 0.9° or 1.8°), 2/ Perimeter length, 3/ Area, 4/ Compaction factor, 5/ Minmax feature, 6/ Endmid feature, 7/ Minmid feature, 8/ Midmax feature, 9/ Distance-to-minpoint feature, 10/ Distance-to-midpoint feature, and 11/ Distance-to-maxpoint feature. These files were first produced on the DUET by the DUMC unit function *toduet()* (AP1.3.2.1.20) working in conjunction with the *'Feature-read-and-store'* program (AP1.4.2.3). From the DUET it was then possible to transfer the feature data to the NUMAC MTS mainframe facility. The purpose written plotting program in (AP1.4.3.2) then allowed the plotting of specified features from this file on the laser printer. Example plots of both identification and orientation features as a function of the item's angular rotation for the file in figure 10.12 can be seen in figures 10.4 to 10.7 and 12.29 to 12.31 for a bowl over a range of 180° and for an equivalent cup file over a range of 360° , in figures 10.8 to 10.11 and 12.32 to 12.34.

The way the feature extraction algorithms are called is that if a single identification feature is not available for an outline, for example if the tracking algorithm has failed to produce a continuous closed outline, then the derivation of subsequent features is stopped and a dummy set of zero features stored in the feature file for the given particular angle entry. In the plots these dummy sets of data are obvious by the dip of the function to zero.

In figure 10.4 the plot of the bowl compaction factor (P^2/A) demonstrates the opposing effects of a slightly increasing shadow perimeter length with a decreasing shadow area, as seen in figures 9.3 and 9.4, between rotation angles 30 and 40 units (54° to 72°). The plot then drops to a minimum as the long sides of the shape become aligned with the incident light direction and the bowl shadow takes up its most 'compact' shape. Small dips in the function are also visible where small corner edge alignment with incident light direction occurs and hence reduces the perimeter

length. When figure 10.4 and 10.8 are considered together it can be seen that, as anticipated the compaction factor is approximately symmetrical about 90° and 270° , although due to edge and tracking effects the plots can hardly be considered mirror images. The 'minmax' feature is also a feature which it would be expected is symmetrical and plots 10.5 and 10.8 show the effect more reliably than the compaction factor. As described in Chapter 9, because of the symmetry of these functions, the 'minmid' and 'midmax' features shown in figures 10.6 and 10.7 for the bowl and 10.10 and 10.11 for the cup are needed for object orientation purposes. It can be seen from the bowl plots in figures 10.6 and 10.7 that, around the angle of rotation of 40–60 units (72° to 108°), valuable orientation information would be lost as edge alignment occurs if just one of these orientation features were to be used. Because of the sections of constant low values about 90° this could possibly lead to an ambiguous object orientation determination. The same features for the cup in figures 10.10 and 10.11 exhibit even stranger asymetrical effects between 0 and 110 units due to the effect the cup handle has on the shadow cast by the cup.

The plots of the bowl pick-up point features in figures 12.29, 12.30, and 12.31 are best explained by cross reference to the plots in figures 12.7b to 12.16b. By considering the distance of the 'minpoint' to the pick-up point centre, marked by the concentric circles, for instance, it can be seen how the feature first dips around 8 rotation units to a value equal to half the major axis. It then returns to a value equal to half the length of a diagonal up to just over 90° , where it drops to a minimum value equal to half the width of the minor axis. As would be expected the maxpoint to pick-up point feature roughly mirrors this behaviour. The midpoint to pick-up point distance, as would be expected, maintains a roughly constant value at around half a diagonal length.

To understand the plots of the cup pick-up point features in figures 12.32, 12.33, and 12.34, one must cross refer to plots in figures 12.17b to 12.26b. It becomes apparent from these that appart from at occasional angular orientations where the cup handle plays a part, for example at around 50 units for the midpoint to pick-up point feature, the values of these features are roughly constant at the radius of the circular part of the cup.

The erratic and irregular behaviour of the features in all of these plots, even for comparitively standard shapes, adds further justification for the use of learned object information in preference to the generation of object features by CAD based

object modelling systems.

The total run time of a program to rotate the table, and capture and transmit an image over a rotation angle of 360° in 0.9° steps is about 15 minutes or about 2.5 seconds per image.

12.4 The Identification and Orientation Program.

Refer to figure 10.20 for a flow diagram of the DUMC unit program. Figure 12.35a is printout of the DUMC unit screen output of this program. By comparing these figures it can be seen where each function is called. Figure 12.35b is the associated line section plot for the program run. The run-time for the analysis of the image processed in figure 12.35b without print and plot outputs is about 3.5 seconds.

In the testing sequence the Apple unit program merely provides a raw binary test image of the tray (refer to flow diagram 10.19 and appendix AP1.3.1.2). It is also possible, as in the learning stage, to input images that have been prestored on DUMC unit disc.

The next section describes what is happening in the DUMC unit program with reference to the printout in 12.35.

In figure 12.35a in the set-up phase first a tolerance of one unit is specified on each of the lower and upper values of each of the feature ranges to allow for small differences in the measured features of objects as compared with the learning stage. Next a number of miscellaneous program options are set including the specfication of the DUMC unit internal disc is as the source of test images. Then the bit-image, feature table and results printout and DUET line section plot options are specified.

In the look-up table generation program the system is set up to identify and orientate the cup, bowl, and plate by specifying the input into the system of the cluster data library files, 'cd1', 'bd1', 'pd1' respectively, created by the cluster program. The files are read in and the clusters mapped into the look-up table.

An indication is given in the printout at this stage of the cluster collisions that are occuring. This is when two or more different clusters are mapping to the same location.

The next stage is the disc image input-parameters stage. This routine allows

one to specify disk files or sequences of disk files for input into the program. In figure 12.35a the base name of the file specified is test_ and the base index is 7. Thus a filename test_7 is generated and the file read in for processing.

First the raw binary image is printed out as specified in the set-up phase. After the edge has been extracted the edge extracted image is then output. Next the feature extraction stages are called and then the identification routine. The printout of this stage shows attempts to resolve significant outlines above the threshold length into their respective object classes.

It can be seen that data set 3 is recognised as a plate, set 4 as a bowl, set 6 as a cup, 26 as a bowl, and 27 as a cup. The system is unable to identify data set 30.

In the next processing stage the intercept points of the radius arcs from the three endpoints are found and then the centre of gravity of the triangle defined by the three genuine intersection points is found. This correponds to the required item pick-up point. A line-section plot of the image is then produced on the DUET-16 micro with the pick-up points marked as a cross (figure 12.35b). The circular item pick-up point markers used in the learning stage were left in place in this testing stage to allow one to see the accuracy of the calculated pick-up points.

A printout of tabulated features was produced next which allowed one to verify the identifications made on the data sets of outlines, in the identification section.

A series of runs was then made on test images as above with separate jumbled items. Printouts of the results of these runs can be seen in figures 12.36a to 12.40a, with their associated plots in figures 12.36b to 12.40b.

The orientation algorithm performance is demonstrated for bowls in plots in figures 12.41 to 12.44. The arcs for the pick-up points are marked for the bowl as it is rotated by 45° in each subsequent image. There are equivalent orientation plots for cups rotated through angles of 54° in figures 12.45 to 12.50.

The next set of figures demonstrates the performance of the identification routine on tray images which had items in close proximity to each other. Figures 12.51 to 12.54 are plots of runs with two items – a bowl and a cup, on the tray. The objects are rotated through 45 degrees on the rotary table between each image. Figures 12.55 to 12.62 are equivalent plots of runs with four items – two cups and

two bowls, in contact and rotated through angles of 45° . The performance of the recognition algorithm in these cases can be judged by if and where the system has placed a pick-up point marking cross on each identified object in the plot of the image. In these examples the rotation of the items on the rotary table has the same effect as altering the incident light direction and suitably modifying the software to accomodate this.

12.5 Discussion of Results.

When the results of using the different thinning algorithms are considered, in the approach chosen to use a single thinning algorithm to recognise separate recyclable items, the choice of the first algorithm seems to offer the most reliable results. This algorithm produces the edges in the positive pixels around the outside of the shadow and preserves object detail. Of the other two algorithms, which both produce edges around the inside of the shadow, the second one often causes the tracking algorithm to fail in producing continuous outlines around each separate shadow. The third one does produce promising results in terms of providing continuous outlines in cases of some nearby objects, but as described earlier in section 12.2.1, does suffer from a loss of image detail.

A possible solution to make use of the best properties of the first and third algorithms would be to use both algorithms separately on the original binary image, and duplicate the feature processing on the on the edge extracted image. A combined approach like this could well significantly reduce problems due to the failure of the tracking algorithm in the most economical way, despite the almost factor-of-two increase in image processing-time overhead.

The decision to discard the smaller tracked line-sections rather than produce objects defined by a number of line sections, or attempt the computationally expensive operation of line-segment recombination, considerably simplifies the software required for image analysis. The possibility of resegmenting smaller incomplete line-sections to ensure that continuous outlines were established, and the use of prior knowledge about objects to segment shadow outlines produced by overlapping objects was considered. Both of these techniques were however decided against on the grounds of algorithm simplicity and speed of operation.

The learning program software seems to perform well in terms of automating

the capture of images and deriving the features from them. The plots of the features derived as functions of an item's angular rotation show both small perturbations and large scale failures of the algorithms at specific angles. The pertubations are seen when the theoretical plot for bowl area in figure 9.3 are compared with the actual bowl area plots in figure 9.5. It can be seen that overall the two results are similar but that small, spikey, ups and downs occur on what should be smooth sections of the actual plot. These are mainly due to digitization effects as a result of low image resolution. When the theoretical bowl perimeter outline plot in figure 9.2, is compared with the actual perimeter plot in figure 9.4 it has the similar basic shape but is not accurate in detail as edge alignment with incident light direction occurs. This difference is mainly due to the simplicity of the theoretical model and the difficulties in easily modelling the digitization effects.

The large perturbations in the actual plots of features, where the values drop to zero, are due to the failure of the tracking algorithm to trace all the way around the outlines of individual shadows. The solutions to this already discussed have included the use of a combination of different thinning algorithms and improved tracking algorithms including the resegmentation of broken outlines. In addition to these methods other ways might be to improve the original image by logically ANDing two or more raw binary images together in the Apple before transmitting them. This would help by reducing the spurious points in the image, that are near the binary threshold and hence 'on' in some images and 'off' in others, as it is often points like these that divert the tracking algorithm. This ANDing operation would take only a small portion of a second and be done concurrently in the Apple with the processing of the last transmitted image. This would hence result in no net increase in the overall image processing time for a sequence of test images.

Another way to reduce the problems of the tracking algorithm would be to increase the resolution of the system to perhaps a 512 × 300 pixel system. This would give an increase in observable image detail and reduce digitization effects. With the given processing power this increase in image resolution would however represent an approximate four-fold increase in processing time.

Additional improvements that could be made to the learning program could include the capturing of two item images at each learning rotation, or using smaller angles of rotation. If two images were captured at each step this would help to take into account the behaviour of points on thin edge sections, which are 'on' in some

images and 'off' in others at the same object position. The overlay of five images of items at 1.8° angles of rotation in figure 12.6 demonstrates how small an angle of 0.9° is. For the given stepper motor used in the project this angle could not be reduced without the use of a gearbox. It is assumed that the way the clusters are produced in the clustering program, as being defined by ranges of features, means that the intermediate rotation angles between each step are automatically taken care of. The only way a smaller rotation angle could improve this is at the angles where the rate of change of features is greatest – that is where large edge alignment occurs, or at orientations where the tracking algorithm fails.

Another improvement that could be made is to attempt to compensate for the distortion of the chosen features when they are placed in different positions on the tray. This could be achieved by having an additional stage in the learning program in which an item is placed in one or more angular orientations at each of the four corners and centre of the tray in the camera's field of view. By interpolating between the feature values at these positions a position-dependant modifying function could be generated for each feature. From experience it was found that the only chosen feature to be significantly affected by an item's position on the tray was the shadow area. The variation of this feature laterally in the x direction is corrected by a linear x displacement dependant function in the recognition stage. The coefficients of this function have been determined empirically by placing a given item at different x positions on the tray and noting the area variation. The function is set up to give a zero area correction with the item at the learning position.

Other alternatives to making geometric corrections to the given features would be to use more parallel light light sources, possibly with some form of light collimation, and to use features which are not significantly affected by an object's position. The features could include for example the number of corners above a certain angle around an outline, or a direction vector frequency distribution.

With regard to the treatment of features, despite the perturbations in the measurement of features the system is 'measurement tolerant'. This is to the extent that if a measurement is made on an item in the learning program, with the item in a particular orientation, and for instance the tracking or feature deriving algorithm behaves strangely, then as exactly the same algorithms are used to derive the features in the recognition program the same behaviour would be repeated for an item in the image in the same orientation and the object would hence be correctly associated

with the learnt object at that orientation.

In the identification program run the printout in figure 12.35a shows the system being set up to recognise the cup, bowl, and plate, by reading in their associated cluster-data library files. The first thing to notice in the printout is the number of cluster collisions which are occuring. These occur in the following situations:–

1/ Where clusters of different items have overlapping features.

2/ Where clusters of the same item have overlapping features due to the 'rectangularization' of cluster features.

3/ Where tolerances are included on the end of cluster feature ranges.

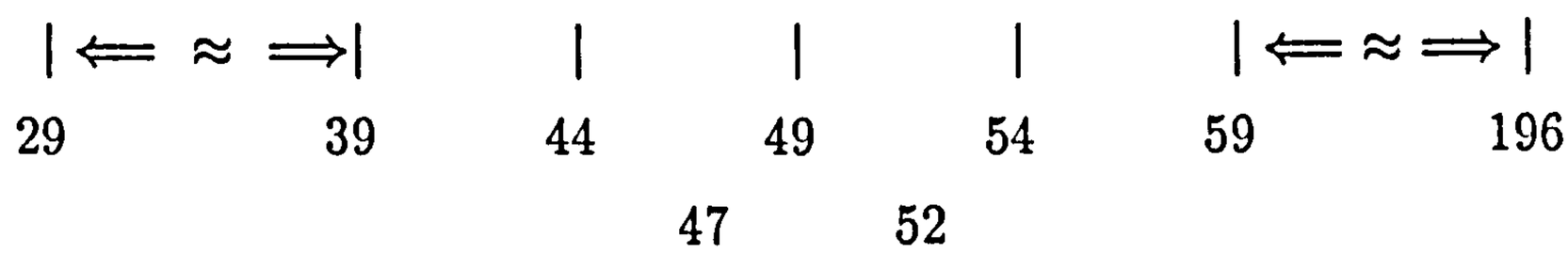
4/ Where feature overlap occurs due to the lack of resolution in lookup table entries.

In the section of the printout indicating cluster collisions, the clusters are coded with an eight bit hexadecimal number. The three most significant bits of this number indicate the number in the order that the library file is read in. In the case of this run the cup file is read in first and is represented by a one, followed by the bowl and plate files as two and three respectively. In the printout these appear as 2, 4, and 6 in the left hand column of the printed cluster code. The five least significant bits of this code correspond to the actual cluster number within the library cluster-data file. Hence for a typical code printout of Hx4e this corresponds to cluster data file 2, – a bowl file, cluster number 14.

It can be seen that cluster collisions of the various types described are occuring in the collision listing. The main factor that has a bearing on most of the collisons is the fourth one listed.

As described in the lookup table production section – Chapter 10.6, the ranges of features over all clusters are scaled to lie in a range of either four or five bits. For example in the sample program run in figure 12.35a the compaction factor ranges from 29 to 196. This range of 167 units is scaled into a five bit (32 valued) address space. Hence in this case each memory address corresponds to a 5 unit feature range. At the top and bottom ends of the ranges this resolution effect may have the effect of increasing the feature range for a given cluster. For example with the above compaction range consider a cluster in which the compaction range is 47

to 52 units. The mapping of the given six point feature range into this space is as follows:-



This means that the range of 47 to 52 has mapped to an expanded range equivalent to 44 to 54 or 11 points. It is this expanded range that causes the overlap of clusters that with a higher resolution would remain separate. The use of tolerances on cluster feature ranges also has the effect of increasing this affect by unpredictably taking ranges over address boundaries. For example a specified one unit of tolerance on the minmid feature above would either cause no effective increase in feature range or three units increase in feature range. Hence the tolerance values on features should be used carefully to avoid unnecessary collisions.

Another related factor to be considered on these lines is that if this system is to be used in a general industrial application where for example a range of parts is presented on a conveyor belt, and there is a wide but not well distributed range of sizes in these parts, then the linear range scaling effect could drastically lower object feature resolution. An possible solution to this would be to have a non-linear scaling of features into the lookup space, depending on the cluster feature distribution. For example if a number of items had features of approximately the same size, with only one or two items with features of significantly different sizes, it would be advantageous to expand the amount of memory used in the lookup table for the features of objects of similar size, in order to improve the resolution of these objects. In common with the cluster generation program this processing would be off-line and would not result in an increase in image processing time.

As the lookup quantization effect described is probably the dominant factor in causing the cluster collisions between clusters of the same item in the printout, then the rectangularization-of-clusters effect must to a large extent be swamped out. In the case of overlapping clusters of different items this must play a part but there is also evidence for overlap due to similarities in measured identification features of clusters of objects to be recognised. For instance all three items are roughly the same size and in the case of the bowl and plate, the long edge of the bowl is of comparable size to the short edge of the plate. Hence for separation of two such

clusters the height difference, and hence shadow area, has to be relied upon. As one might expect this is not always sufficient. This is indicated for example by the overlap of bowl cluster 3 with plate cluster 4 in the cluster collision list of figure 12.35a.

As suggested in Chapter 10 there are a number of ways of changing the system programs to improve the performance of the system as a whole. These could include deriving an extended range of features in a learning stage including for example more geometric measurements, the number of corners, higher moments of inertia, and direction vector frequencies, and then using a form of covariance analysis on these in the clustering program to determine the best features to differentiate clusters. There could also be associated penalty functions in this selection for ease of extraction and computing time cost. This task could be performed on a 'per item' basis or on an 'over all items' basis, although the task is complicated by not knowing the object classes in advance. The feature choice information could then be passed on disk into the identification program which would then derive only the features necessary to identify the specified objects.

When the results in figures 12.35a to 12.40a and their corresponding line section plots in figures 12.35b to 12.40b are considered, it can be seen that the identification program performs well. In figure 12.35a and 12.35b all items have been correctly and uniquely identified. In figures 12.36a to 12.40a it can be seen that a range of identification errors has occured.

In figures 12.40 it can be seen that the result in the case of sample number 5, corresponding to the cup in the top right corner, the values of the features have lead to the access of a lookup table address containing no cluster index marker. This hence means the system is unable to make any identification of the object. Other misidentifications of objects occur when an object maps to a location with a cluster collision marker in. This indicates an ambiguity of identification and can be seen in figures 12.36, 12.37, 12.38, 12.39 and 12.40. As an example in figure 12.36 the cup in the centre of the tray has been ambiguously classified as a cup and a bowl and in figure 12.38 the bowl in the top left hand corner has been ambiguously classified as a bowl and a plate. Using the chosen identification features these ambiguities are unresolvable at this stage.

The only case where an actual wrong cluster identification has occured in in figure 12.37 where a bowl has been misidentified as a plate – in the bottom left

hand corner of the image.

All of the above identification errors can be accounted for by:-

1/ Cluster collisions as a explained above.

2/ Changes in rig conditions since the learning operation, for example tray illumination, binary capture threshold, and changes in camera-object distance or zoom setting.

3/ Geometric distortion of features depending on the position of the object on the tray.

Solutions to all of these particular problems have already been discussed.

Figures 12.35, 12.37, 12.38 and 12.39 demonstrate the ability of the algorithms to identify objects below other objects in circumstances where their shadows do not interfere. In these figures it can be seen where the cup has been placed upon the plate and both items have been identified.

In all these plots in figures 12.36 to 12.40 in the case of ambiguous identification the orientation algorithm uses the pick-up point radii for the first of the corresponding cluster data files entered. This allows an ad-hoc ambiguity resolution to be used in that the item cluster-data library files are input in order of decreasing probability of finding an item on the tray. For example the plate library file is input last because there is only ever one plate to be found on the tray whereas there are two bowls.

The plots in figures 12.41 to 12.50 show the performance of the orientation algorithm on images of bowls rotated through angles of 45° and cups through 54° . It can be seen how the pick-up point radius arcs are associated with each identified object. The centre-of-gravity routine then determines the centre of the correct three intersect arcs to give the pick-up point which is then identified in the plot by a cross. The performance of the pick-up point finding algorithm on sample images can be seen by observing the crosses in figures 12.35b to 12.40b. To have an indication of the accuracy of this routine the pick-up point markers used in the learning stage have been left on the items. The radius of the large outer circle is 14 mm. and of the inner circle is 4 mm.

The pick-up point finding routine in these examples is working from cluster-data library files for which an accuracy of 7mm. was specified in pick-up point

location for cluster generation. It is clear from the plots that the algorithm using three arcs finds such points well within this accuracy, often with an accuracy of 4mm. or less.

The final sets of images looked at are worst case situations for the algorithm working on debris-free items on the flat tray. These plots demonstrate the performance of the identification algorithms on items in close proximity. An indication of an object identification is given by the presence of a pick-up point marker cross in these plots. For figures 12.51 to 12.54, with just a single bowl and cup being rotated together at angles of 45° on the rotary table it can be seen that the objects are each correctly identified and have their pick-up points calculated in two of the four images. The way the shadow of the item not identified is affected, for example in figure 12.52, suggests many possible sources of failure of the identification routine. These may be anticipated to include a non-identification or mis-identification due to disturbed shadows. The chosen cluster mapping method in the latter case offers the advantage being more likely to make a rejection of such an object than would a Euclidean feature distance cluster matching method which would in such a case attempt a probably erroneous best fit. If a robot were being used to manipulate the objects so identified then the wrong identification could easily translate into unsafe handling and damage to robot or item.

Further examples of items in close proximity can be seen in plot figures 12.55 to 12.62 in which the images contain four items – two bowls and two cups, with each item surrounded by three others and the whole group rotated through 45° steps on the rotary table.

It can be seen that the cups and bowls are correctly identified in various images, but for the most part the tracking algorithm has severe difficulty in extracting outlines corresponding to single items. With three surrounding items the effect that items in close proximity have on neighbouring item's shadows is far more pronounced.

The reason for rotating the items is to demonstrate the effect of changing the incident light direction on a fixed tray. With two or more light sources switched on in sequence and images captured at each switching, with a minor modification to the software, the image could be processed to give a much improved identification performance. The easiest way to effect the change in the existing software used to identify objects to process images illuminated from a different direction would

be to perform a rotation operation on the input image array. For example for a 180° change in incident light direction the only changes necessary would be to first perform a mirror operation on the initial image and then, at the pick-up point calculation stage, mirror the pick-up point coordinates back to get the coordinates in the image frame.

Although it appears that a multiplication in the number incident light directions potentially increases the ability of the system to recognise items that are in close proximity to other items it can be seen by considering the eight figures from 12.55 to 12.62 that an identification of one item per scene cannot be guarenteed. Although over these eight figures all the items have been correctly identified in at least one image the identification rate is well below an average of even two items per scene over eight images. If random debris were included this figure could be expected to drop further. In addition the correct identifications are accompanied by numerous misidentifications, – due to overlapping and disturbed shadows.

If all these factors are considered together it must be concluded that in this near worst-case, overlap situation the recognition program does not perform well.

The main computing overhead of a system using many incident light directions is in the multiplication of images to be processed. For images similar to the ones in the test images this is about 3.5 seconds per image. There would also be some additional software required to arbitrate between identifications and pick-up point locations arizing from different images of the same tray.

With the existing binary vision system, despite the above limitations this approach seems to offer the best potential for the long term objective of the recognition of overlapping items and items near to other items, which may be obscured by random debris, as well as considerably enhancing the performance of the system on the restricted development project problem. The processing time of 3.5 seconds for the single image system developed is well within the target time of 5 seconds and around the maximum anticipated robot operating speed in performing the unloading operation on a single vision system per robot cell. The viability of the system operating on multiple images would depend on the amount of extra time available and/or the additional cost of increasing the performance of processing hardware for example by duplicating the central processor unit or including dedicated special-purpose hardware to perform the simple tasks like the thinning and edge extraction operation.

12.6 Second Project Conclusions.

The results show that when the system is set up to operate on the development problem of learning, recognising, and finding the pick-up points of the cup, bowl and plate arranged separately in any rotational orientation, upright and flat on a base tray, requiring no skilled external supervision, the system performs the task well. The image testing program provides a reliable identification and pick-up point location well within the target processing time of five seconds, and to an accuracy often greater than an estimated required accuracy of 7mm. Errors that do occur in the performance of the program are largely as a result of correctable changes in rig environment between learning and recognition stages. These often involve changes in the illumination conditions, and compensatable-for distortions of features across the tray due to the illumination geometry. Other errors are due to the lack of memory available for the lookup table and the physical similarity of object classes of different items as described by the chosen features. The system could hence be expected to perform even more reliably with small numbers of more dissimilar items.

The performance of the system working on a single image of items in close proximity shows that it could not be depended on to make reliable identfications in this situation. The possibility of using multiple light sources switched on in sequence offers promising possibilities for object identifications in more general scenes – although at a significant processing-time penalty.

The approach adopted, using conventional hardware, to an extent relies on 'brute force' to achieve its high speed of object identification. This is in the sense that it capitalizes on the low cost of conventional microprocessor memory to make the look-up table in the identification stage, with most of the work required for recognition being performed off-line in the learning and clustering stages.

As the system stands, the only restriction on what can be recognised in the given time is the amount of memory available to it for the look-up table. This means that instead of merely increasing on-board system memory by adding a few megabytes of RAM chips the potential offered to the system by the use of a relatively cheap 100Mbyte hard disc would allow the use of five features with a resolution of five bits. If this system were coupled to an intelligent feature selection stage, perhaps using multivariate analysis, then the potential is for a low-cost extremely versatile

self teaching vision system. It would in principle be possible to produce such a system suitable for a wide range of industrial robotics and sorting applications by attaching a low cost binary image capture front end to a standard 'off-the-shelf' microprocessor with hard disk at a total hardware cost of less than £2000.

Suggestions for the extension of the work done in this project include the use of increased hard disc memory, the development of intelligent feature selection techniques as described above, and the use of more elaborate non-linear cluster mapping techniques. In addition an investigation into the use of more suitable yet low cost graphics processors like the Texas Instruments TMS34010 in the place of the existing processor, and the use of low cost dedicated hardware to perform simple repetitive operations such as smoothing and thinning on the image, could be expected to greatly enhance the performance of the system. In order to realize the enormous potential of such a system all that would remain to be done would be to survey the wide range of potentially suitable industrial applications and tailor the system to meet their requirements.

APPENDIX 1

THE COMPUTER PROGRAMS

AP1.1 Introduction.

The Apple Microcomputer software is written in a mixture of Applesoft Basic and Motorola M6502 assembly language. The slower running supervisory Basic programs make calls to the assembly language routines at known addresses.

The DUMC unit software is written in the 'C' programming language. It is structured in a modular form so that the subfunctions performing different operations on the image can be called independantly by a main coordinating function. This means that functions can be substituted or removed completely without affecting the flow of the program. The only departures from this structure are the inclusion of calls to the library input/output utility functions which are necessary to output characters via the serial ports to the VDU screen, the dot- matrix printer, and the DUET-16 microcomputer.

In the DUMC unit program passing parameters between functions is done through the use of external variables. This is because of the large number of common variables needed in the program and also the pass by argument nature of 'C'. This means that parameters are passed between functions as copies of the original and that sub-functions are unable to modify the original variables. This can be achieved by using a more involved process of pointer passing but for the number of variables required this would have unnecessarily complicated the program.

AP1.2 Programs for Project One.

The following sequences represent a typical program run on the Apple and DUMC unit systems.

1/ The Apple Supervisor Program.

1.1/ – Takes an ideal image.
1.2/ – Takes another ideal image.
1.3/ – Calls the 'AND' subroutine.

1.4/ – Takes a test image.
1.5/ – Calls 'EOR' subroutine.
1.6/ – Calls transmit subroutine.

2/ The DUMC Unit Supervisor Program *(main())*.

2.1/ – Calls option modification routine – *modithr()*.
2.2/ – greetings routine – *greet()*.
2.3/ – image reception routine – *aquip()*.
2.4/ – image expansion routine – *bitmatpw()*.
2.5/ – optional image printing routine – *pripwb()*.
2.6/ – chosen pixel counting routine – *ufcf()*.
– *ufcoupw()*.
– *ufcw()*.
2.7/ – decision making routine – *decpw()*.

The implementation of each of these functions is explained in more detail in the following descriptions.

AP1.2.1 Apple Unit Programs.

AP1.2.1.1 Apple supervisor program.

Refer to figure AA1.

This program first sets up the digitizer and other input/output devices by writing to their various command registers. An ideal image is then produced by calling the image capture routine twice and 'ANDing' the two images together with the 'AND' assembly language subroutine called at 24630 (Hx6036). A test image is then taken, the image 'EORed' with the ideal, by calling the subroutine at location 24576 (Hx6000). The resulting image transmitted by the transmission subroutine at 720 (Hx2d0). The use of these assembly language routines effects a much more rapid processing than could be achieved by using the Apple Basic language.

AP1.2.1.2 'AND' and the 'EOR' subroutines.

Refer to figure AA2.

These routines both work by applying the relevant microprocessor instruction to coincident points in the areas of Apple graphics memory corresponding to the images. For the high resolution pages used these start at Hx2000 and Hx4000

and extend for Hx2000 bytes. The routines are mainly concerned with the setting up of the correct addresses of the relative coincident bytes of the graphics memory to logically operate on.

AP1.2.1.3 The transmission routine.

Refer to figure AA3.

This routine is also concerned with setting up the correct addresses of the bytes of screen memory to be accessed for transmission. In order to reduce the program complexity of the image reception routine in the DUMC unit the screen memory is read out in a raster fashion starting at the top left corner. Because of the irregular mapping of the Apple high resolution graphics memory the addresses are part loaded from a look-up table at locations Hx03a0 and Hx03c0. The data is read out by writing it to the data output register on the parallel interface card.

AP1.2.2 DUMC Unit Programs.

AP1.2.2.1 *main()*

Refer to figure AD1.

This is the main coordinating function of the program which calls the sub-functions in series. The function first declares and initializes all the external variables. Inside the function itself a number of local variables for controlling the looping of the main program are declared as are the functions returning variables. The sub-functions are then called in the following sequence at the end of which the user is given the option of re-running the whole sequence.

AP1.2.2.2 *amodithr()*

Refer to figure AD2.

This function allows the user to select options to print the image subsequently received from the Apple, choose the particular area counting routine to be used, and modify the identification thresholds and count windows.

AP1.2.2.3 *greet()*

Refer to figure AD3.

This function makes repeated calls to the library formatted print statement

to output an introductory banner onto the console VDU.

AP1.2.2.4 *aquipwap()*

Refer to figure AD4.

This function first initailizes the parallel interface by calling the *ipar()* library routine. It tests for an 'X' start character in the input stream and then makes further repeated calls to the library parallel interface read function *getpar()*, to read in an 8K byte binary image from the parallel port. These bytes are stored in the continuous array 'screen'.

AP1.2.2.5 *bitmatpw()*

Refer to figure AD5.

This function converts the 8K byte binary image in the array 'screen' into the expanded 54K one bit per byte array 'mpixel'. This is achieved by bitwise masking of the original array elements by a rotating bitmask.

AP1.2.2.6 *ufcf()*

Refer to figure AD6.

This is one of the three similar, selectable counting routines. It counts the number of 'on' pixels within the chosen counting windows and outputs the counts for each on the VDU screen. The two other routines *ufcoupw()* and *ufcw()* are essentially the same but display more information on the screen.

AP1.2.2.7 *decpw()*

Refer to figure AD7.

This function takes the counts in the windows from the previous stage and compares them with their respective thresholds. If the counts are below the thresholds an indication of an item's absence is printed on the VDU screen else the item is given as being present.

AP1.3 Programs for Project Two.

AP1.3.1 Apple Unit Programs.

There were two Apple programs developed for the second project. The first one described is the one used in conjunction with the DUMC unit item learning program and is described in the flow diagram figure 10.2. The second Apple program described is the single image capture and transmit routine used with the image testing routine –flow diagram figure 10.19.

AP1.3.1.1 The Apple item learning program.

Refer to figure AA5.

In this program the section – lines 100–330 initializes the hardware and introduces the program. If the set-up phase is required a jump is made to the set-up routine at line 1000 and the user prompted to adjust the rotary table position and the binary capture threshold. After this the rotation parameters are input between lines 350 and 500. At this point the main sampling loop is entered in which the stepper motor is rotated, the images are captured, and the raw binary image transferred. The transfer routine called at Hx0720 is described in the first project description AP1.2.1.3.

AP1.3.1.2 The test image capture program.

Refer to figure AA4.

This Apple supervisor program acts in a similar way to the supervisor program in the first project described in AP1.2.1.1. It differs in that, instead of calling the AND and EOR subroutines to act on a captured image, it transmits the raw captured binary image as above.

AP1.3.2 DUMC Unit Programs.

The complete vision system comprises three separate DUMC unit programs: the item learning program, the cluster-data library file creating program, and the image testing program. Figures 10.3, 10.16, and 10.20 respectively are flow diagrams of these programs.

AP1.3.2.1 The item learning program.

AP1.3.2.1.1 *main()*

Refer to figure AD8.

The main function in this program acts like the *main()* function in the first project. It first declares and initializes all the external variables and data structures required in the program. Within the function *main()* the program makes calls to the hardware initialization routines *cache()* (appendix AP1.4.1.1) and *initdu()* (appendix AP1.4.1.2). It next calls the greeting routine *greet()* (AP1.3.2.1.4) and then the general program parameter setting routines which prompt the user to specify program options. These routines are *changep()* (AP1.3.2.1.2) and depending on whether the image source is specified as being from internal disk or Apple, *filepar()* or *apar()* (AP1.3.2.1.3) respectively. It then enters the main program loop. First it reinitializes those external variables re-used in processing each image and then calls the following subroutines in sequence:

1/ a) *aquipwd()*
   b) *aquipwap()*
2/ *bitmatpw()*
3/ *kon()*
4/ *selshad()*
5/ *fclosed()*
6/ *xend()*
7/ *fperi()*
8/ *farea()*
9/ *compact()*
10/ *shadw()*
11/ *cimp()*
12/ *ccircle()*
13/ *dcentre()*
14/ *timp()*
15/ *toffile()*

The following optional utility routines can be called:

16/ *pripwb()*
17/ *printable()*
18/ *shipcornpw()*
19/ *toduet()*

AP1.3.2.1.2 *changep()*

Refer to figure AD9.

This function simply gives the operator the option of modifying the general program parameters. These are self explanatory in AD9.

AP1.3.2.1.3 *filepar()* and *apar()*

Refer to figure AD10 and AD11.

filepar() is the routine which prompts the user to input the parameters of disc image storage files to be read in in sequence. *apar()* prompts the user to specify the number of Apple images to be sampled.

AP1.3.2.1.4 *greet()/ aquipwap()/ bitmatpw()*

These are the same or similar to functions in the first project (AP1.2.2.3), (AP1.2.2.4), and (AP1.2.2.5) respectively.

AP1.3.2.1.5 *aquipwd()*

Refer to figure AD12.

This function first generates a filename from the specified base filename by incrementing a number at the end of the filename string. The image file with this name is then read into the array 'screen'.

AP1.3.2.1.6 *kon()*

Refer to figures AD13a and AD13b.

These functions produce the thinned, edge extracted outlines from the expanded binary image. Three alternative functions were developed. The one finally adopted and the first one described here is written in 'C', and produces a shadow outline around the outside of the shape (figure AD13a).

If points in the image are found to be 'on' with at least one of their four-connected neighbours 'off', that is digital zero, the point is allowed to remain 'on' – unless none of the surrounding points is 'on'. If this is the case the point is labelled 2 in the array, so as not to affect subsequent processing of the same array. The final stage converts these digital 2 elements to zeroes.

In processing the image first the four corners are tested then the top and

bottom rows, the left and right columns and then the central block of the image.

The second algorithm, described here for competeness, produces an outline around the 'inside' of a shadow of pixels that were originally digital zero (figure AD13b) and is written in M68000 assembly language. The algorithm given is the third one described in Chapter 8. As in the algorithm just described first the four corners are tested, the top and bottom rows and the first and last columns. Finally the central image block is tested. At each stage the logical conditions described in chapter 8 are applied to pixels surrounding the point under test.

AP1.3.2.1.7 *selshad()*

Refer to figure AD14.

This function scans the edge extracted image from left to right down the image. The scan starts from point after the last line starting point scanned or the top left corner when the scanning is first started and looks for a digital 'one' point. Once one has been found a jump is made to a track section of the program which then follows a chain of 'one' points labelling each tracked point with a distinct line code number. The tracking continues until no further points can be found. The tracking then stops and the user is returned to the main function with an indication of whether the full image has been scanned. If it has not the function is immediately recalled. This process is repeated until the whole image has been scanned.

Labelling each point as it is found removes it from the array and prevents its subsequent re-tracking on the same line. This would happen for example, if the line were in a loop. The line extracted is coded into a continuous array containing all the coded lines in order to save space and for processing speed. Separate line-number-indexed pointer arrays are used to point to the starts and ends of these lines, and a unique end character marks the ends of each of these lines in the line array.

In the line array points are stored sequentially with their i and j coordinates, followed by their direction coding. Then follows a point's sharpness coefficient which indicates the degree of change in tracking direction at a point. The order in which the lines are found determines the line number index for subsequent line feature arrays.

AP1.3.2.1.8 *fclosed()*

Refer to figure AD15.

This function determines whether a shadow outline is closed. This is achieved by first testing the x proximity and then the y proximity of the start and end points and deciding whether they are below a defined threshold. If an outline is found to be closed it is labelled as such within the line indexed array 'type'.

AP1.3.2.1.9 *xend()*

Refer to figure AD16.

This function provides the artificial ends of lines used to ensure that the features derived from subsequent chain coded lines are accurate. The function first tests whether the line is completely closed with adjacent start and end points. If it is not, whichever end has the lowest x-coordinate is followed back along the line until it has the same x-coordinate as the other end. This provides the new alternative starts and finishes to the line in two arrays which are then used in subequent chain code processing of the lines.

AP1.3.2.1.10 *farea()*

Refer to figure AD17.

This function calculates the area of the shadow region within the closed outline. The modified start and endpoints are used in order to ensure the accuracy of the area calculated by the function.

In order to avoid the addition of halves and the need for their subsequent computationally expensive floating point multiplication the column area elements are first doubled, by a computationally economical bitwise left shift, before adding to the running area total. Finally this total is divided by two by a rightwise bit shift to give the true area which is stored in a line indexed area array.

AP1.3.2.1.11 *fperi()*

Refer to figure AD18.

This function calculates the perimeter lengths of lines by adding up separately all a line's even and odd direction vectors and doing a single final multiplication, addition and division. This way costly computing operations such as any

floating point operations, multiplications, and especially divisions are minimized.

The resulting perimeter length is stored in a line indexed perimeter array.

AP1.3.2.1.12 *compact()*

Refer to figure AD19.

This function provides the compaction factor $C = P^2/A$ by indexing the area and perimeter arrays produced by previous functions and performing the necessary multiplications and divisions on them. The sequential indexing operation is performed efficiently using pointers. The resulting compaction feature is again stored in a line indexed array.

AP1.3.2.1.13 *shadw()*

Refer to figure AD20.

This function produces the three significant points on the inside of the edge. First there is a test to see if the shadow corresponds to a shadow cast inside the particular tray item. If it is the processing is terminated. If not the program continues with a test to determine whether the inside edge of the shadow is tracked first or whether the tracking first goes down the edge furthest from the light source. If this is the case the program jumps to suitable processing which deduces the points from the line working in reverse and starting with the last point. In both cases conditions are applied to points on the line to find the three required points. All of this processing is performed efficiently using pointers, and the resulting coordinates are stored in a line number indexed array.

AP1.3.2.1.14 *wlcimp()*

Refer to figure AD21.

This function takes the three points calculated by the *shadw()* function and calculates a series of features which it stores in the array 'cimpoint'. These features are mainly a range of interpoint distances.

AP1.3.2.1.15 *ccircle()*

Refer to figure AD22.

This routine finds the pick-up point circle by considering an outline's bound-

ary length and area. It then finds the minimum and maximum x and y points on the perimeter of the circle and finds the circle centre.

AP1.3.2.1.16 *dcentre()*

Refer to figure AD23.

This routine finds the distance of the pick-up point from the minpoint, midpoint and maxpoint of what is the only significant outline in the image in the learning stage. The routine works by finding the outline, and then taking the circle centre found after this to calculate the required distances.

AP1.3.2.1.17 *timp()*

Refer to figure AD24.

This routine takes the sets of image features from each program run and stores them in an image indexed array 'timpoint'.

AP1.3.2.1.18 *toffile()*

Refer to figure AD25.

This routine prompts the user to specify a name for the feature data storage file. The feature data for all the images processed is output by calls to the disk operating system.

AP1.3.2.1.19 *pripwb()*, *printable()* and *shipcornpw()*

These functions are described in the utility subroutine section AP1.4 in sections AP1.4.1.3, AP1.4.1.4, and AP1.4.1.5, respectively.

AP1.3.2.1.20 *toduet()*

Refer to figure AD26.

This function is similar to *toffile()* except that the feature data is written out on one of the serial lines to be stored on disc file in the DUET-16 microcomputer. This enabled the data to be subsequently transferred to a mainframe facility for further processing.

AP1.3.2.2 The item cluster-data file generation program.

AP1.3.2.2.1 *main()*

Refer to figure AD27.

This acts as the main supervisor program calling the following functions first in the sequence:–

1/ *cache()*
2/ *initdu()*
3/ *greet()*
4/ *cluscrit()*
5/ *fromfile()*
6/ *standardize2()*
7/ *crit1()*
8/ *meansdnp2()*
9/ *varmaxt1()*
10/ *clusplit()*
11/ *clusprint()*
12/ *move2()*
13/ *crit2()*
14/ *varmaxt2()*
15/ *detinfo()*
16/ *storfile()*

AP1.3.2.2.2 *cache(),initdu(),and greet().*

These functions are similar to or the same as AP1.4.1.1, AP1.4.1.2 and AP1.2.2.3 respectively.

AP1.3.2.2.3 *cluscrit()*

Refer to figure AD28.

This routine prompts the user to enter the external criteria for cluster production.

AP1.3.2.2.4 *fromfile()*

Refer to figure AD29.

This routine prompts the user to specify the feature data file to be read in. It then reads in the file stored in a special format by the routine *tofile()* (AP1.3.2.1.18) by calls to the disc interface routine *system()*, with different parameters. The data

is stored in the clusinfo type structure 'c1' in the array 'featvars'.

AP1.3.2.2.5 *standardize()*

Refer to figure AD30.

This function takes all the read in feature data in the array 'c1.featvars' and calculates the totals of each feature, the mean of each feature, the sum of the squares of the distance from the mean of each feature and the standard deviation of each feature. Each original feature measurement then has the mean of that feature subtracted from it and the result divided by the standard deviation. The resulting standardized feature data is then stored in the array 'c1.sfeatvars'.

AP1.3.2.2.6 *crit1()*

Refer to figure AD31.

This function tests the current number of clusters against the initial requirement for the number of clusters and returns a value indicating whether the required number has been reached.

AP1.3.2.2.7 *meansdnp2()*

Refer to figure AD32.

This function calculates the mean and variance of each cluster.

AP1.3.2.2.8 *varmaxt1()*

Refer to figure AD33.

This function indexes through all clusters and all four identification features to find the cluster and feature with the maximum variance.

AP1.3.2.2.9 *clusplit()*

Refer to figure AD34.

This routine takes the cluster with the feature with the largest variance – whose cluster index is passed into the function in the variable 'invarmax' and whose feature index in the variable 'fvarmax', and finds the maximum and minimum of this feature for this cluster. It then splits the cases in this cluster into two clusters by testing to see which of these extremes the cases are nearest to for this feature. If

it is nearer the minimum value the case is assigned to a new cluster. If not it is left in the original cluster.

AP1.3.2.2.10 *clusprint()*

Refer to figure AD35.

This is a VDU outputting utility subroutine to print out the members of all of the existing clusters.

AP1.3.2.2.11 *move2()*

Refer to figure AD36.

This routine moves cases to the cluster whose centre is nearest. The routine first calculates the feature means of all clusters and then for each case calculates the distance to each cluster centre. The minimum cluster centre distance for each case is determined and if the distance is not a minimum for the cluster the case already belongs to it is moved to the nearest.

AP1.3.2.2.12 *crit2()*

Refer to figure AD37.

In this function the second clustering criteria is tested. The maximum and minimum of each pick-up point distance is found for each of the clusters. The cluster with the maximum range of pick-up point feature is then found and this maximum compared with an allowed possible maximum in the variable 'mnrange' set by the user in *cluscrit()* (AP1.3.2.2.3). A value is returned indicating whether or not this is greater than the required range.

AP1.3.2.2.13 *varmaxt2()*

Refer to figure AD38.

This function takes the cluster index of the cluster with the maximum pick-up point range calculated in *crit2()* above, and first tests the total variance in the given cluster. If this total is zero this means that the routine to find the variance may have done so inaccurately and in this case the feature with the maximum range is labelled for splitting. Otherwise the feature with the maximum variance for the given cluster is labelled for splitting.

AP1.3.2.2.14 *detinfo()*

Refer to figure AD39.

This routine determines the following parameters for each cluster:-

1/ The bisector of each pick-up point range.

2/ The case labels in each cluster.

3/ The maximum and minimum of each identification feature in each cluster.

AP1.3.2.2.15 *storfile()*

Refer to figure AD40.

This routine allows the user to specify a cluster data disk storage file in which to store the above data. At the start of the routine library header file information about the file is entered. By a sequence of system calls to the disk system interface this header information and the above cluster data is written out to disk.

AP1.3.2.3 The image testing program.

AP1.3.2.3.1 *main()*

Refer to figure AD41.

The function *main()* acts in the same way as the *main()* in the other DUMC unit programs. It first declares and initializes the external variables and data structures for use in the program. Within the function the variables needed on each image processing run are reinitialized and calls are made to subroutines for the first time in the following order:-

1/ *cache()*
2/ *initdu()*
3/ *greet()*
4/ *changep()*
5/ *lookup()*
    1/ *rdbkdata()*
    2/ *clscreen()*
    3/ *map()*
    4/ *maptest()*
6/ a) *filepar()*

b) *apar()*

7/ a) *aquipwd()*

b) *aquipwap()*

8/ *bitmatpw()*

9/ *kon()*

10/ *selshad()*

11/ *fclosed()*

12/ *xend()*

13/ *shadw()*

14/ *cimp()*

15/ *fperi()*

16/ *farea()*

17/ *arcorrect()*

18/ *compact()*

19/ *identify()*

20/ *circept()*

21/ *cockhat()*

With optional output utility routines:

1/ *printable()*

2/ *shipcornpw()*

3/ *shipcen()*

AP1.3.2.3.2 *cache(),initdu(),greet(),changep().*

These functions are the same as or similar to those in the item learning program AP1.4.1.1, AP1.4.1.2, AP1.2.2.3, and AP1.3.2.1.2.

AP1.3.2.3.3 *lookup()*

Refer to figure AD42.

This function is the coordinating subfunction which sequentially calls the functions necessary to generate the look-up table.

AP1.3.2.3.4 *rdbkdata()*

Refer to figure AD43.

In this function one is prompted to specify the cluster data-files corresponding to items to be recognised in the testing sequence. The files are read in and first

their header information, for example item type, is printed out on the VDU screen. This is followed by a printout of the actual cluster data.

AP1.3.2.3.5 *clscreen()*

Refer to figure AD44.

This is a utility function which takes each read-in cluster and for each variable produces a horizontal bar graph showing where the cluster feature ranges come. This allows one to see where possible cluster collisions might occur.

AP1.3.2.3.6 *map()*

Refer to figure AD45.

This routine produces the cluster look-up table from the read-in cluster data. In order to do this first the maximum and minimum of each feature is found over all the clusters read in. The range is then taken and the scaling factors calculated to make this range between 0-31 for the features compaction and minmax, and 0-15 for the features minmid and midmax. These factors are stored in the array 'div'. The minimum of each feature is then taken as an offset.

For each item file, for each cluster, for each identification feature, the memory in array 'map' is mapped out with the continuous ranges of cluster variables. The integer address offset into the array is made up with the five least significant bits corresponding to the scaled compaction feature, the next five least significant bits corresponding to the scaled minmax feature, the next four least significant bits the scaled minmid feature, and the next four least significant bits the midmax feature. This gives a maximum possible address offset from the base of 2^{18} or 250 K short words. The array 'map' is an array of short words, two bytes long. The lower byte is used to characterize the the cluster being mapped. The five least significant bits of this byte denote the cluster number and the most significant three bits the file number. A unique code of Hxf f is inserted into this byte if in the mapping sequence it is found that the location has already been accessed. In this case the upper byte is used to give an index into a cluster collision array in which the colliding clusters are stored in byte form as above. Before a cluster collision array entry is formed from the colliding clusters, a test is made to see if this particular cluster combination exists. If it does the index of the existing cluster is stored in the map instead.

AP1.3.2.3.7 *maptest()*

Refer to figure AD46.

This is a utility routine to test the look-up table. The user enters a set of identification features for an object from the keyboard and is able to see whether a correct identification is made.

In the function the features are scaled and manipulated so that they generate an index into the look-up table in exactly the same way as features derived from an object in the test image would.

AP1.3.2.3.8 *filepar(), apar(), aquipwd(), bitmatpw(), kon(), selshadpw(), fclosed(), xend(), shadw(), cimp(), farea(), fperi(), compact()*

These functions are the same as those descibed in the item learning program AP1.3.2.1.3 through to AP1.3.2.1.14.

AP1.3.2.3.9 *arcorrect()*

Refer to figure AD47.

This routine is implemented in the testing program to apply a linear x position dependant correction to the measured shadow area. The x position of the shadow is approximated by taking the x coordinate of the minpoint. The scaling factor applied to this has been found empirically by area measurements on the shadow of an item in the same orientation but in different x positions across the tray. The correction is adjusted to be zero when the object has an x coordinate of 145 – the position of the item when it is learnt in the learning stage.

AP1.3.2.3.10 *identify()*

Refer to figure AD48.

This routine takes each outline with a set of features and first tests the range of each feature to ensure that it is not outside the permitted range for objects to be identified. The function then constructs the index of an address offset into the look-up table from the features in the same way as the addresses are generated in the *map()* (AP1.3.2.3.6) and *maptest()* (AP1.3.2.3.7) routines. The bitwise rotations ensure that the address generated keeps the features orthogonal to each other. The particular location pointed to by this address is accessed and the contents tested.

As described in the mapping function if the value of the contents is zero the object is not recognisible. If there is a value in the lower byte of the word not equal to collision index the item file and cluster number are reconstructed from the value and these used to access other item cluster information, including the cases in the cluster and their ranges of angular orientation.

If the lower byte of the addressed element is equal to the collision marker the upper byte is used as the access into the collision array. The details of the colliding clusters are then output.

AP1.3.2.3.11 *circept()*

Refer to figure AD49.

This routine takes the three radius arcs and the three conditional points – the minpoint, midpoint and maxpoint, for each identified object and calculates the position of the six intersections for the three cicles with these radii and these centres. The corners of crosses centred on these intersection points are then calculated and stored in the array 'cross'.

AP1.3.2.3.12 *cockhat()*

Refer to figure AD50.

This routine takes the sets of six intersections to find the true intersecting centre. For both the midpoint-minpoint and the midpoint-maxpoint radius intersections the true intersection points are taken to be the ones for which the x coordinate is smallest. This comes from the negative square-root solution of the quadratic in *circept()*. The bisector of the line joining both these two negative solutions and the nearest minpoint-maxpoint arc intersection point is found. These two points – the bisector and the nearest minpoint-maxpoint intersect are taken as the end points of a line from the bisector of the side of a triangle to the vertex of a triangle defined by the three true intersection points. The centre of gravity of this triangle is then calculated as being two thirds of the way along this line from the vertex. This point represents the best available approximation of the true pick-up point.

AP1.4 Utility Programs and Subroutines.

AP1.4.1 DUMC Unit utilities.

The routines *cache()* and *initdu()* are hardware initialization routines for the DUMC unit. The other routines were developed to facilitate program development. None of them is necessary for the flow of the main program but they were often vital in determining program parameters and essential in debugging operations.

AP1.4.1.1 *cache()*

Refer to figure AD51.

This is an assembly language routine which starts the cache running on the DUMC unit Motorola 68020 processor. The instruction to start the cache was not available on the M68000 assembler used and so the op-code was entered as data in the routine which would be treated as an instruction when encoutered by the processor.

AP1.4.1.2 *initdu()*

Refer to figure AD52.

This function contains a number of serial interface routines to initialize and output characters from two of the DUMC unit serial interface ports.

AP1.4.1.3 *pripwb()*

Refer to figure AD53.

This is an optional routine which if called takes the image as stored in the expanded image array 'mpixel', and performs the necessary manipulations on the data to produce a series of image bytes suitable for the vertically orientated dot-matrix printer head. These bytes are then output to one of the DUMC unit's serial printer ports, via calls to the *hwrite2()* library subroutine. The port is connected to an Epsom FX100 dot-matrix printer.

AP1.4.1.4 *printable()*

Refer to figure AD54.

This optional function provides a complete printout of the range of features

calculated for each line above the given threshold value on the dot-matrix printer.

The first part of the function is concerned with setting up the printer with the correct control codes. This is accomplished by repeated calls to the serial output routine *hwrite2()*. Another library function *decode()* was used to convert the numerical variables into the character strings suitable for outputting. These strings are then output followed by the printer default reset codes using the serial character output routine.

AP1.4.1.5 *shipcornpw()*

Refer to figure AD55.

This is an optional utility which outputs the coordinates of tracked line corners to the DUET-16 microcomputer for plotting. The DUET has equivalent reception software for plotting the line on the screen between the corners and storing it on disc. The corners are determined by points on a line which have a sharpness coefficient greater than zero corresponding to a change in tracking direction. The function sequences through all the lines. It starts by sending a predefined start character followed by the first point on the line. The x and y coordinates are sent as two consecutive shorts which, because *hwrite3()* expects a pointer to characters of eight bits, if told to output four characters with a pointer to the first of two shorts of 16 bits the two shorts are sent each in two separate parts most significant byte first.

The function then goes through testing each successive point for a sharpness coefficient greater than zero. If one is found the coordinates are output as described before.

The final character is sent irrespective of its sharpness , followed by the stop character. This stop character is chosen as 254. In order to prevent this being sent accidentally each coordinate value is tested before it is output. If a coordinate is found with this value its value is changed to 253 which makes negligible difference in the final plot. The use of this stop character removes the need for a more complex transfer protocol involving indicating the size of block to be transferred before coordinate transmission takes place.

AP1.4.1.6 *shipcen()*

Refer to figure AD56.

This routine outputs on the serial line the coordinates of the corners of crosses centred on the derived identified object centres to the DUET-16 microcomputer for plotting in the same way as described in *shipcornpw()* above.

AP1.4.2 DUET utilities.

A number of utilities were developed to run on the DUET-16 microcomputer. The DUET was used for its clear high resolution graphics presentation and plotter driver facility, and the ability to write data disks in a suitable form to allow transfer onto a mainframe facility. Because the plotting facilities were remote from the development station the plotable image was stored by the project DUET on disc and transfered to a remote DUET for plotting. The two Basic language routines written to do this are described in the following two sections. The routine developed to store the feature information provided by the output routine *toduet()* (AP1.3.2.1.20) was written in 'C' and is the third program listed here.

AP1.4.2.1 *The Read-in-and-Store program.*

Refer to figure ADU1.

This program is used when the DUET is connected by RS232 serial link to the DUMC unit. The DUMC unit routine *shipcornpw()* (AP1.4.1.5) or *shipcen()* (AP1.4.1.6) outputs a series of line section corners in coordinate form down this serial line. The DUET program first requires the user to specify a filename into which the corner coordinates are to be stored. The program after receiving a start character then proceeds to read in the corner coordinates of lines with each line separated by a stop character indicating the end of the line. The coordinate values are read in as two separate 8 bit numbers, MSB first, and combined and stored in the chosen disc file. On each successive stop character a draw subroutine is called to draw the line section just stored by plotting between successive corners. The process is repeated until all the image has been transferred.

AP1.4.2.2 *The Disc-Read-and-Plot program.*

Refer to figure ADU2.

This program is run with the DUET-16 connected to a Hewlett-Packard graphics plotter by RS232 serial line. The program after first inviting the user to specify the file to be plotted, outputs the control characters required to draw a diagramatic framework for the plot. It then recovers the series of x and y corner coordinates from the file and outputs them in a form suitable for the plotter to plot between them. For clarity of presentation of the start of each line, plotting colours are changed in sequence with each separate line. The program also plots the line sections on the DUET monitor screen in the equivalent colour.

AP1.4.2.3 *The Feature-Read-and-Store program.*

Refer to figure ADU3.

This routine is the feature data reception routine to run in conjunction with the feature data output routine *toduet()* (AP1.3.2.1.20). It consists of a main function which makes calls to the various serial input and interface library functions. The routine starts by including the particular libraries containing the necessary interface functions. In the main loop the user is prompted for a filename in which to store the feature data. The program tests for a start character and when this is found starts reading in the data. The first data following the start character tells the program how many feature data sets are to be read in.

The program reads in the specified number of data sets and then outputs these onto disc for storage. A disc written on this system has an IBM compatible format and can readily be transferred to the mainframe using standard file transfer facilities.

AP1.4.3 NUMAC mainframe programs.

These programs were written in Fortran on the NUMAC MTS system at Durham to make use of the GHOST80 graphics interface and laser printer facility on this system.

AP1.4.3.1 *The Bowl Area and Perimeter predicting routine.*

Refer to figure MF1.

In this routine one can optionally select one of two area or two perimeter predicting routines. Having made the selection one is invited to input the required number of side lengths, the angles of the corner sections as required, and the height of the particular item. One is then invited to enter title strings for the plot and its position. The required area or perimeter calculations are then made and the data output via the graphics interface to the plot generator.

AP1.4.3.2 *The feature-data file plotting routine.*

Refer to figure MF2.

Before running this program feature data files for the bowl, cup and plate items are copied into the files NB1, NC1, and NP1, respectively. When the program is run one enters an option for the type of item whose features are to be plotted. The corresponding file is then read in. One then enters the options for the particular features to be cross-plotted. Option 1 at this stage corresponds to the angular rotation index in units of 0.9° or 1.8°. After this one is invited to input a heading and scales for the plot. The plot is then generated by calls to the graphics library interface routines.

APPENDIX 2

THE COMPUTATIONAL STEPS OF THE MAINFRAME

BMDP K-MEANS CLUSTERING PROGRAM

These steps constitute a resumé of the steps in the BMDP mainframe clustering program as described at the back of the BMDP manual. For a more comprehensive description of this program and its use refer to the manual – reference [65].

1/ Standardize the data by the overall variance or covariance if so requested.

2/ Classify cases into clusters based on the SEED or CENTRE parameters, if they have been stated.

3/ If the desired number of clusters have been defined, then go to step 5 otherwise, split a cluster into two clusters via the following procedure:–

a) for each variable i, in each cluster c, the variance, Var_{ic}, is estimated.

b) the indices i^* and c^* are found, such that $Var_{i^*c^*}^{**} = \max(Var_{ic})$

c) cluster c^* is split into two clusters, based on the value of variable i^*, at the midpoint of the range of variable i^* in cluster c^*.

4/ If the number of clusters is less than desired, repeat step 3.

5/ Standardize the data by the within clusters variance or covariance matrix, if so requested, using a computation similar to that of step 1.

6/ Move each case to the cluster whose centre is closest to the case.

7/ Recompute centres of clusters (means) and repeat step 6 if any case was moved in step 6.

8/ Repeat from step 5 if any case was moved since the last standardization.

9/ Report clusters for the current number of clusters.

10/ If a larger number of clusters has been requested, repeat from step 3.

11/ Produce final report and save data, if requested.

REFERENCES.

1/ R. Nelson and J.R. Corby, 'Machine Vision for Robotics' (I.E.E.E. Transactions on Industrial Electronics, Vol. 1E-30, No. 3, August 1983) pp.282–291.

2/ T.Owen, 'Strategic issues for automated production: the challenge of robotics and computer integrated manufacturing.' (Cranfield Press, Cranfield, Bedford, England 1985).

3/ F.Wilkie, 'Industry opens its eyes to machine vision.' (Process Engineer (GB), Vol. 67, No. 1, Jan. 86) pp.41–43.

4/ T. Prior and W. Pastorius, 'Applications of Machine Vision to Parts Inspection and Machine Control in the Piece Part Manufacturing Industries' Diffracto Ltd. Windsor, Ontario Canada.

5/ Douglas Jones, 'Sighted Robots Will Wear White Collars' (Materials Handling and Automation, The Engineer Survey, 16 April 1981) pp.54–58.

6/ A.Abramovich, 'Advanced vision technology for printed circuit inspection.' (Proceedings of Vision '85 conference, Detroit, MI, USA, 25–28 March 1985) pp.5/168–179.

7/ J.Ragland, 'Automating Innerlayer Inspection.' (Circuits Manufacture (USA), Vol. 25, No. 2, 1985) pp.91–92.

8/ T. Sasaki, Y. Hasegawa, and K. Ozawa. 'Application of self- propelled devices to in-piping inspection technology.' (Robot (Japan), No. 51, March 1986) pp.98–103.

9/ D.J. Braithwaite, 'Vision Guided Robots in the Foundary Environment.' (Proceedings of ISATA 85: In Pursuit of Technical Excellence. International Symposium on Auto- motive Technology and Automation. Graz, Austria. 23-27 Sept., 1985.) Vol. 1, pp.365–74.

10/ Editor 'Vision Systems as Standard ?' (The Production Engineer, May 1983) pp.46.

11/ Don. Braggins, 'Automated Visual Inspection – the Technology Exists' (The Production Engineer, March 1983) pp.21–22.

12/ Don. Braggins, 'An Eye for Inspection' (The Production Engineer, May 1983) pp.31–33.

13/ M.F. Russo, 'Three-D robot vision for automotive glass insertion.' (Proceedings of Sensors '85 Conference, Detroit, MI, USA, 5–7 Nov. 1985) pp.MS85-1010/1–14.

14/ B. Gardstam, 'Car body assembly with ASEA 3D-Vision.' (Proceedings of 5th. International Conference on Robot Vision and Sensory Controls -RoViSec 5. Amsterdam, Netherlands, Oct. 29–31, 1985). pp.509–16.

15/ D. German, 'Vision aided inspection techniques for the automotive industry.' (Conference proceedings of Vision '85 Detroit, MI, USA. March 25-28, 1985). pp.6/1–7.

16/ Tech. Tran. Corporation of America, 'Machine Vision Systems – a Summary and Forcast' 1983. Available from:–

Tech Tran Corporation,
134, N. Washington Street,
Naperville, Illonois 60540,
U.S.A.

17/ H.K. Nishihara, 'Practical real-time imaging stereo matcher.' (Optical Engineering, 23(5), September-October 1985) pp.536–545.

18/ Editor 'On-line 3-dimensional Measurement' (The Production Engineer, May 1983).

19/ B.K.P. Horn, 'Understanding image intensities.' (Artificial Intelligence, 8(2), 1977), pp.201–231.

20/ D.H. Ballard and D. Sabbah, 'Detecting Object Orientation From Surface Normals.' (Pattern Recognition, Vol. 13,1981) pp.111.

21/ Lothar Rossol, 'Computer Vision in Industry' (Proceedings of 1st. Inter-

national Conference on Robot Vision and Sensory Controls, 1981) pp.11–18.

22/ S.W. Holland, L. Rossol and M.R. Ward, 'Consight I: A Vision Controlled Robot System for Transferring Parts From Belt Conveyors' (Computer Vision and Sensor Based Robots, eds. G.G. Dodd and L. Rossol, Plenum Press NY) pp.264–268.

23/ A. Rozenfeld, 'Machine vision for industry: concepts and techniques.' (Robotics Today (USA), Vol. 7, No. 6, Dec. 1985) pp.19–22.

24/ Julius T. Tou, 'Knowledge Based Image Interpretation' (I.E.E.E. Proceedings of International Conference on Cybernetics and Society, Seattle, Washington, USA 1982) pp.515–519.

25/ R.A. Brooks, 'Model based 3-D interpretations of 2-D images.' (IEEE PAMI 5(2): March 1983) pp.140–150.

26/ D.M. Mckeown Jr., W.A. Harvey Jr., J. McDermott, 'Rule Based Interpretation of Aerial Imagery.' (IEEE PAMI 7(5) Sept. 1985) pp.570–585.

27/ B. Julesz and J.R. Bergen, 'Textons, the fundamental elements in preattentive vision in the perception of textures.' (Bell System Technical Journal, 62(6), July– August 1983). pp.1619–1644.

28/ C. Torras and F. Thomas, 'Vision systems for industry.' (Automation and Instrumentation (Spain), Vol. 19, No.154 1985) pp.91–100.

29/ Igor Aleksander, 'Extension of Robot Capabilities Through Artificial Vision: A Look into the Future.' (Digital Systems for Industrial Automation, Vol 2, No. 3).

30/ Koulopolous C. 'Automated Vision for Identification of Silhouette Shapes with an Application to Industrial Automation. City University, Ph.D thesis 1982.

31/ B.W. Kernighan and D.M. Richie, 'The 'C' Programming Language.' (Prentice-Hall Inc., 1978).

32/ R.C. Bolles, R. Horaud, and M.J. Hannah, '3DPO: A 3-D part orientation system.' (Proceedings of the 8th International Conference on Artificial Intelligence, Karlesruhe, West Germany, August 1983.) pp.1116–1120.

33/ S. Tsuji and A. Nakamura, 'Recognition of an Object in a Stack of Industrial Parts.' (Proceedings of 4th Int. Conf. on Artificial Intelligence 1975) pp.811–818.

34/ P.A. Nagin and B. Schwartz, 'Approaches to image analysis of the optic disc.' (Proceedings of the 5th. International Conference on Pattern Recognition, Miami Beach, Florida, USA. 1-4 December 1980) pp.948–56.

35/ J. Mantock, A.A. Sawchuk, T.C. Strand, 'An optical processor applied to cloud classification.' (Proceedings of Society of Photo-Optical Instrumentation Engineers USA, Vol. 208, Devices and Systems for Optical Signal Processing, Los Angeles CA. USA, 6–7 February 1980) pp.167–71.

36/ J.T. Tou and R.C. Gonzalez, 'Pattern Recognition Principles.' Addison-Wesley Publishing Company, 1974.

37/ R.M. Haralick, 'Digital step edges from zero crossings of second directional derivatives.' (IEEE PAMI 6(1), January 1984) pp.58–68.

38/ Azriel Rosenfeld, 'Computer Vision for Industrial Applications' I.E.E.E. Pattern Recognition Vol.16 1983) pp.47.

39/ M.A. Fischler and H.C. Wolf, 'Linear Delineation.' (Proceedings of IEEE Conference on Computer Vision and Image Processing, Washinton D.C., June 19–23, 1983) pp.351–356.

40/ J.B. Burns, A. Hanson, and E. Riseman, 'Extracting Straight Lines.' (Proceedings of DARPA IU workshop, New Orleans, Louisiana USA, October 1984) pp.165–168.

41/ Azriel Rozenfeld, 'Connectivity in Digital Pictures.' (Journal of Association of Computing Machinery, Vol. 17, No 1, January 1970).

42/ A. Rozenfeld and J.L. Pfaltz, 'Sequential Operations in Digital Picture Processing' (JACM, Vol. 14, No. 4, October 1966) pp.471–494.

43/ H. Freeman, 'On the Encoding of Arbitrary Geometric Configurations.' (IRE Transactions on Electrical Computers, June 1961).

44/ G.P. Dineen, 'Programming Pattern Recognition.' (Proceedings of the

Western Joint Conference on Pattern Recognition, Los Angeles 1956).

45/ S.H. Unger, 'Pattern Detection and Recognition' (Proceedings IRE, October 1959) pp.1737–1752.

46/ C. Arcelli, L. Cordella and S. Laviadi, 'Parallel Thinning of Binary Pictures' (Electronic Letters 1, 7 , 1975).

47/ A.P. Reeves and A. Rostampour, 'Shape Analysis of Segmented Objects Using Moments' (Proceedings of IEEE Conference on Pattern Recognition, 1981) pp.171.

48/ P.W. Kitchin and A. Pugh, 'Processing of Binary Images' (Chapter 2, Robot Vision, Ed. A. Pugh, IFS Publications Ltd., 1983).

49/ D.T. Lawton, T.S. Levitt, C. McConnell and J. Glickman, 'Terrain models for an autonomous land vehicle.' (Proceedings IEEE International Conference on Robotics and Automation, San Francisco, California, USA, April 7–10, 1986) pp.2043–2051.

50/ J.K. Tsotsos, 'Knowledge organisation and its role in representation and interpretation for time-varying data: the ALVEN system.' (Computational Intelligence (USA), 1985) pp.1:16–32.

51/ Makoto Nagou, 'Control Stratagies in Pattern Analysis' (Proceedings of 6th Int. Conference on Pattern Recognition, Munich, Germany, October 1982) pp.996–1006.

52/ M.A.Arbib, K.J. Overton and D.T. Lawton, 'Perceptual Systems for Robots' (Interdiscplinary Science Reviews, Vol 9,No. 1, 1984) pp.31–46.

53/ P.S. Chao and P.A. Frick, 'Application of Artificial Intelligence to Robot Vision'(Proceedings Northcon 83 electronics show and convention,Portland,Oregon, USA. May 1983) pp.8/5/1–5.

54/ B.L. Bullock, D.H.Close et al.,'Image Understanding Application Project: Implementation Progress Report' (Computer Vision Systems Vol.27, 1983) pp.50–56.

55/ T.M. Strat and M.A. Fischler, 'One-eyed stereo: a general approach to modeling scene geometry.' (IEEE PAMI 8(6), November 1986) pp.730–741.

56/ J. Ross Stenstrom, 'Syntactic Pattern Recognition for Robot Vision' (Proceedings of the Int. Conference on Robotics, Atlanta, GA, USA, March 1983).

57/ A.P. Pentland, 'Perceptual organisation and the representation of natural form.' (Artificial Intelligence, 28, May 1983) pp.293-331.

58/ R.C. Bolles, R.A. Cain, 'Recognising Partially Visible Objects: the Local Feature Focus Method' (Robotics Research, Vol.1, No.3, Massachusetts Institute of Technology, 1982).

59/ R.C. Bolles 'Robust Feature Matching Through Maximal Cliques' (Proceedings of SPIE's Technical Symposium of Imaging Applications for Industrial Inspection and Assembly, Washington D.C. April 1979).

60/ C. Goad, 'Special purpose progamming for 3-D model based vision.' (Proceedings of Image Understanding Workshop (USA), June 1983) pp.94-104.

61/ J.A. Losty and P.R. Watkins, 'Computer Vision for Industrial Applications.' (The GEC Journal of Research, Vol 1, No.1, 1983) pp.24-34.

62/ Texas Instruments, 'Leading Electronics Press Coverage." (Compiled magazine articles on semicoductor solutions for bit-mapped graphics, Texas Instruments, 1986).

63/ R.C.Gonzalez and R. Safabakhsh, 'Computer Vision Techniques for Industrial Inspection and Robot Control' (Computer 15(12)) pp.17-32.

64/ J.A. Hartigan, 'Clustering Algorithms.' John Wiley and Sons, 1975.

65/ W.J. Dixon (Chief ed.), 'BMDP Statistical Software.' University of california Press, Berkley 1983.

66/ Estabrooke, 'A mathematical model in graph theory for biological classification.' (Journal of Theoretical Biology 12, 1966) pp.297-310.

67/ T. Pun and D. Coulon, 'Visual Prosthesis' (Computer Graphics and Imaging, Vol.15, 1981) pp.210.

BIBLIOGRAPHY.

1/ Ed. Gardner W.E. 'Machine-aided Image Analysis.' Institute of Physics, 1978.

2/ Young T.Y. and Calvert T.W. 'Classification Estimation and Pattern Recognition.' American Elsevier Publishing Co. 1974.

3/ Tou J.T. and Gonzales R.C. 'Pattern Recognition Principles' Addison-Wesley Publishing Company, 1974.

4/ D.A. Rosenthal 'An Inquirey Driven Vision System Based on Visual and Conceptual Hierarchies.' UMI Research Press, 1981.

5/ Ed. Sklansky J. 'Pattern Recognition.' Dowden Hutchinson and Ross Inc. 1973.

6/ D. Marr, 'Vision' Freeman, San Francisco USA, 1981.

7/ Ed. Winston P.H. 'The Psychology of Computer Vision.' McGraw Hill, 1975.

8/ Ed. M.A. Fischler and O. Firschein, 'Readings in Computer Vision.' Morgan Kaufmann Inc. 1987.

9/ Ed. Pugh A. 'Robot Vision' IFS Publications Ltd. 1983.

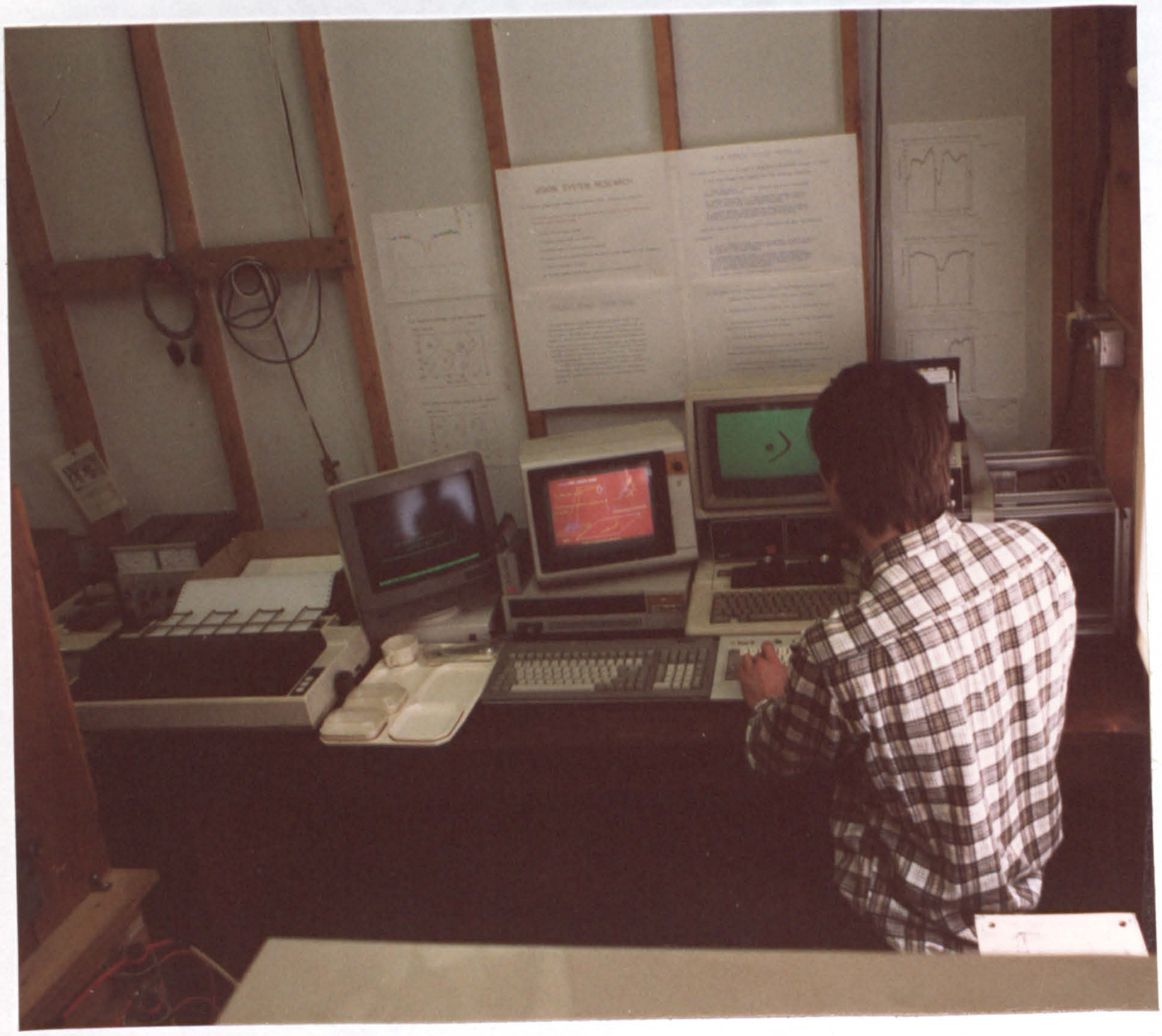
10/ B.W. Kernighan and D.M. Richie, 'The 'C' Programming Language.' (Prentice-Hall Inc., 1978).

11/ Cronshaw A.J. 'Language and Applications Software for an Automatic Vision System to Recognise industrial Components.' University of Nottingham, Ph.D thesis 1981.

12/ Koulopolous C. 'Automated Vision for Identification of Silhouette Shapes with an Application to Industrial Automation. City University, Ph.D thesis 1982.

13/ D.H. Ballard and G.M. Brown, 'Computer Vision,' Prentice- Hall, 1982.

14/ Open University television program made in conjunction with the Alvey Directorate, 'The Alvey Image Processing Project,' broadcast 11:59, January 30th, 1988.

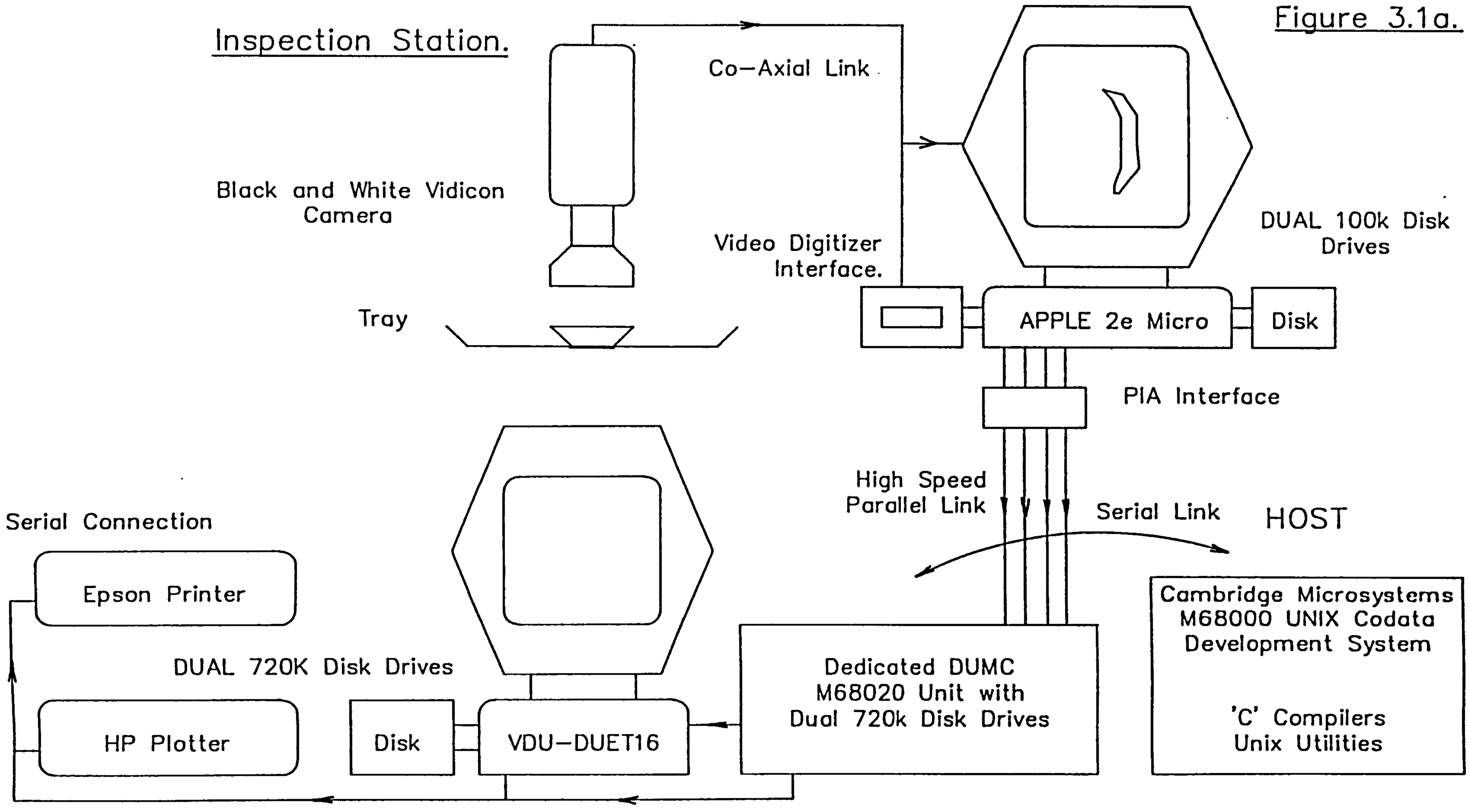


Combined Microcomputer Vision Workstation.

Thesis Figures

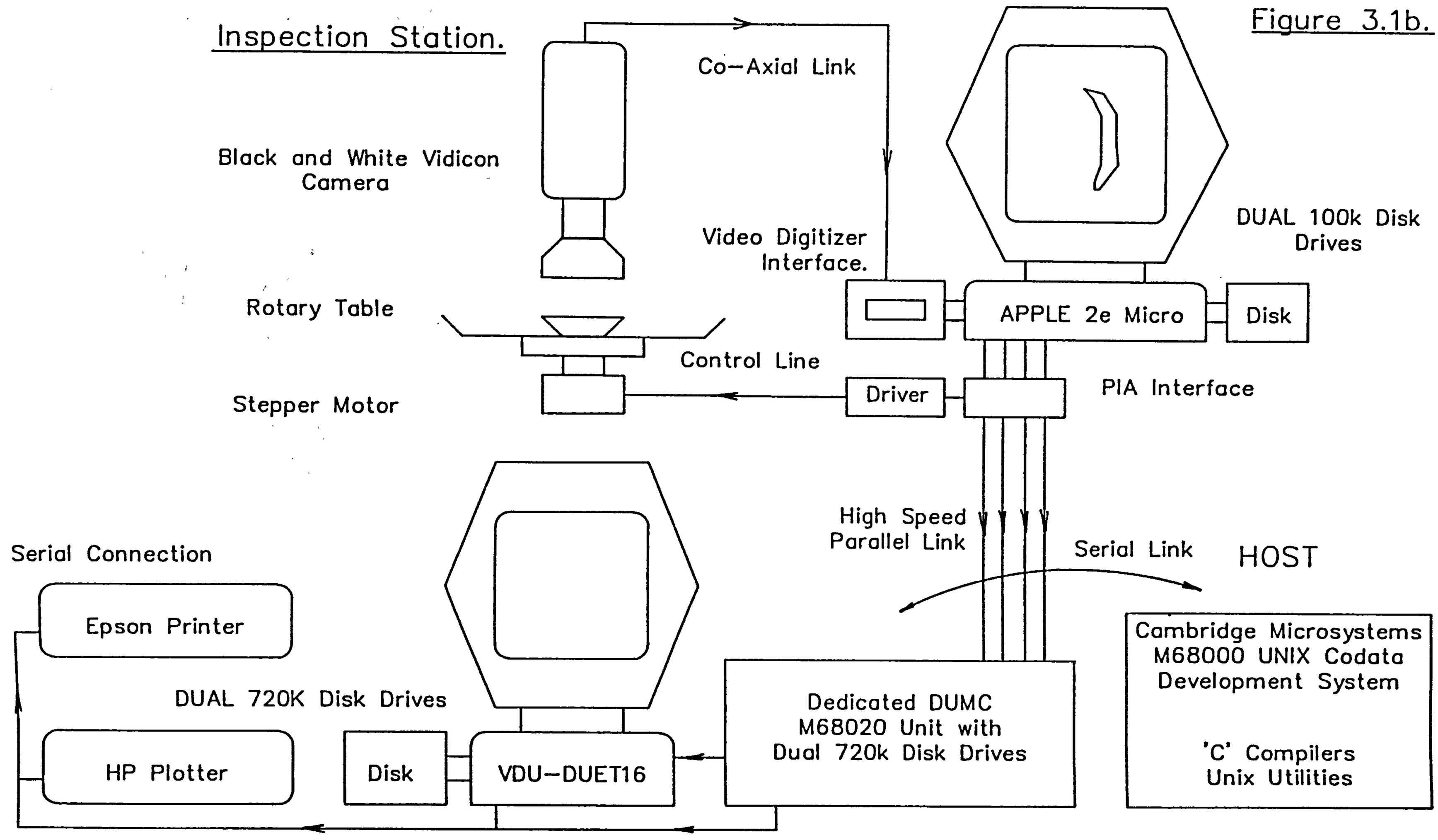
SCHEMATIC DIAGRAM OF VISION SYSTEM HARDWARE FOR PROJECT ONE.

Figure 3.1a.



SCHEMATIC DIAGRAM OF VISION SYSTEM HARDWARE FOR PROJECT TWO.

Figure 3.1b.



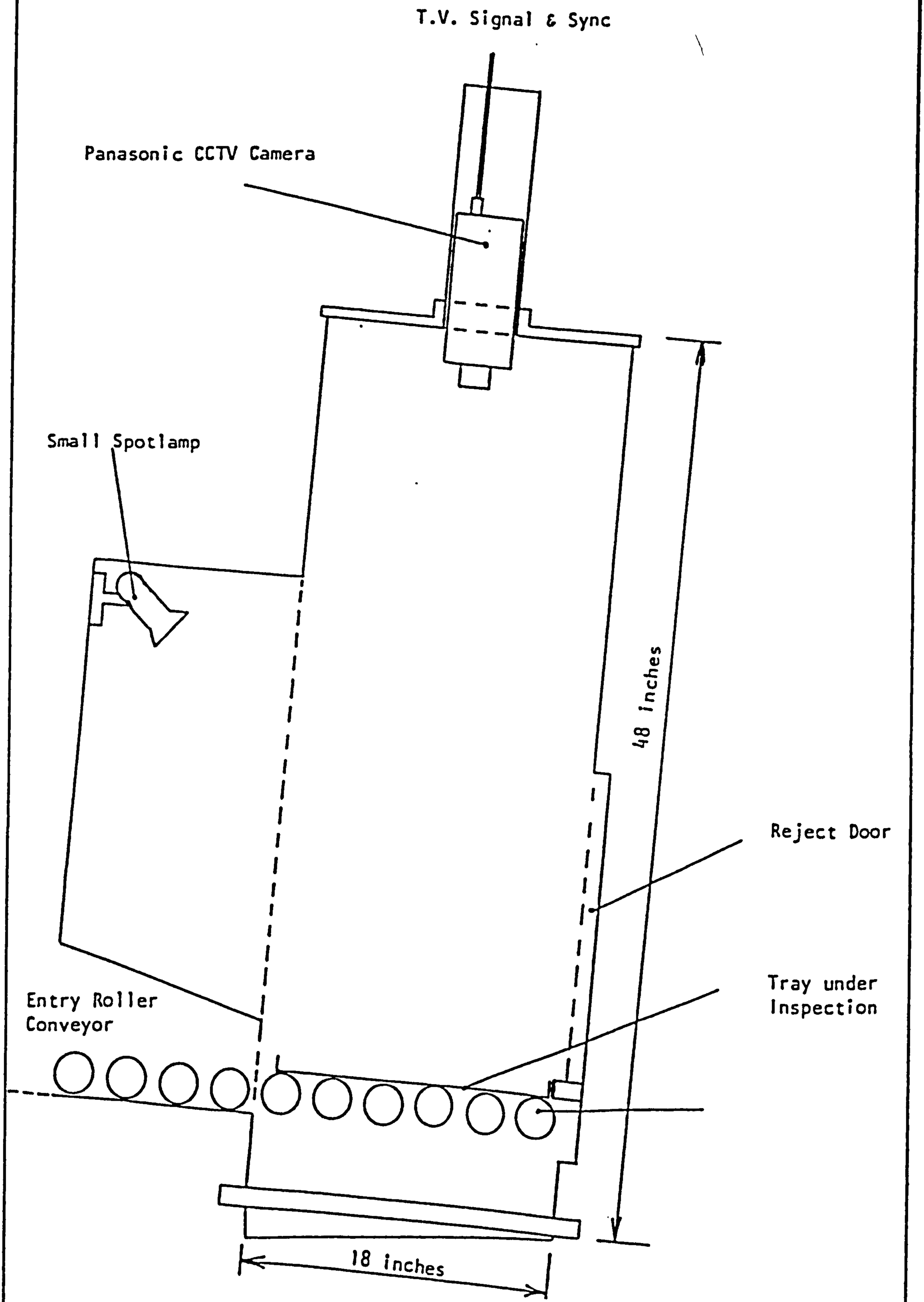
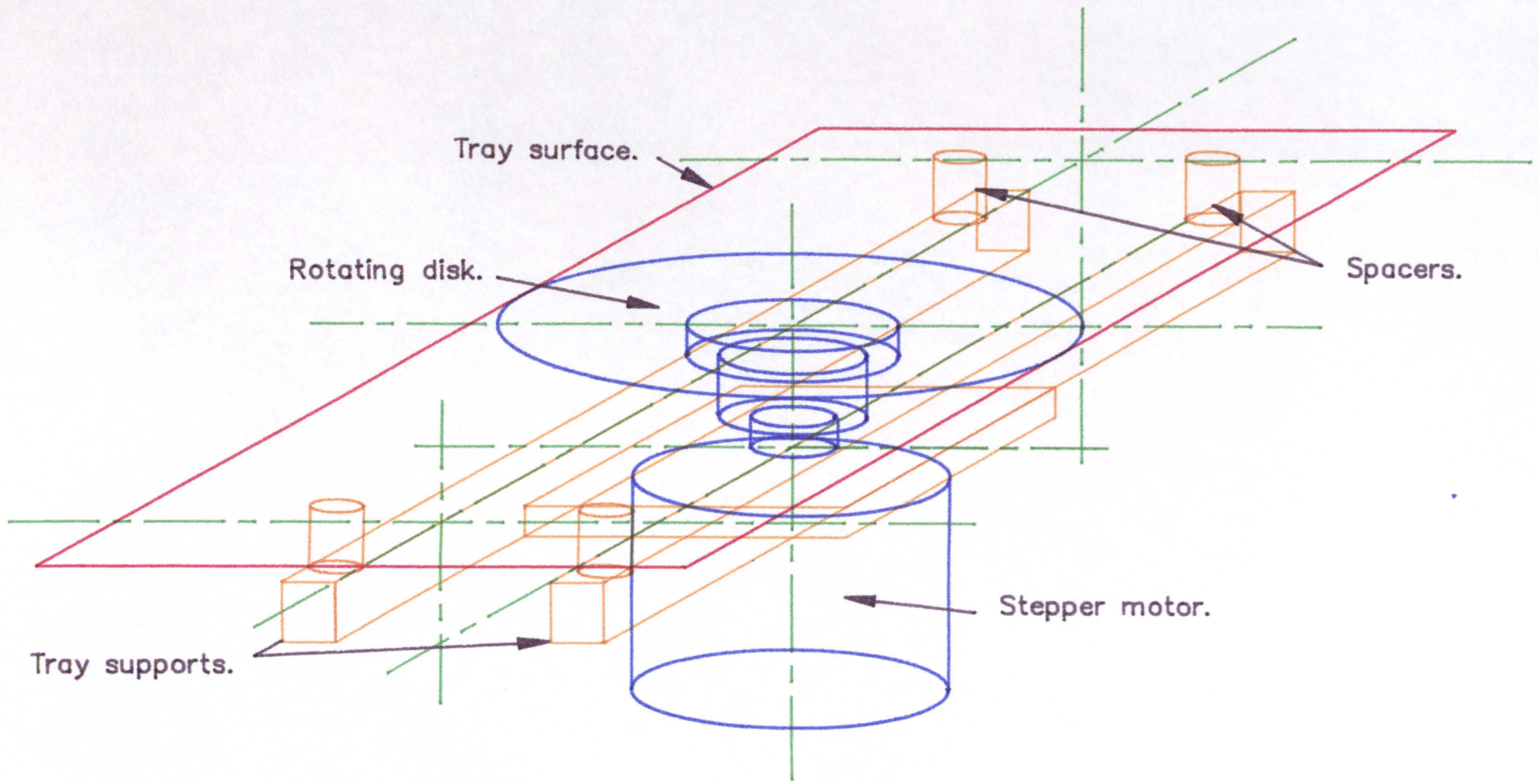


Figure 3.2

LIGHTBOX - CAMERA ARRANGEMENT.

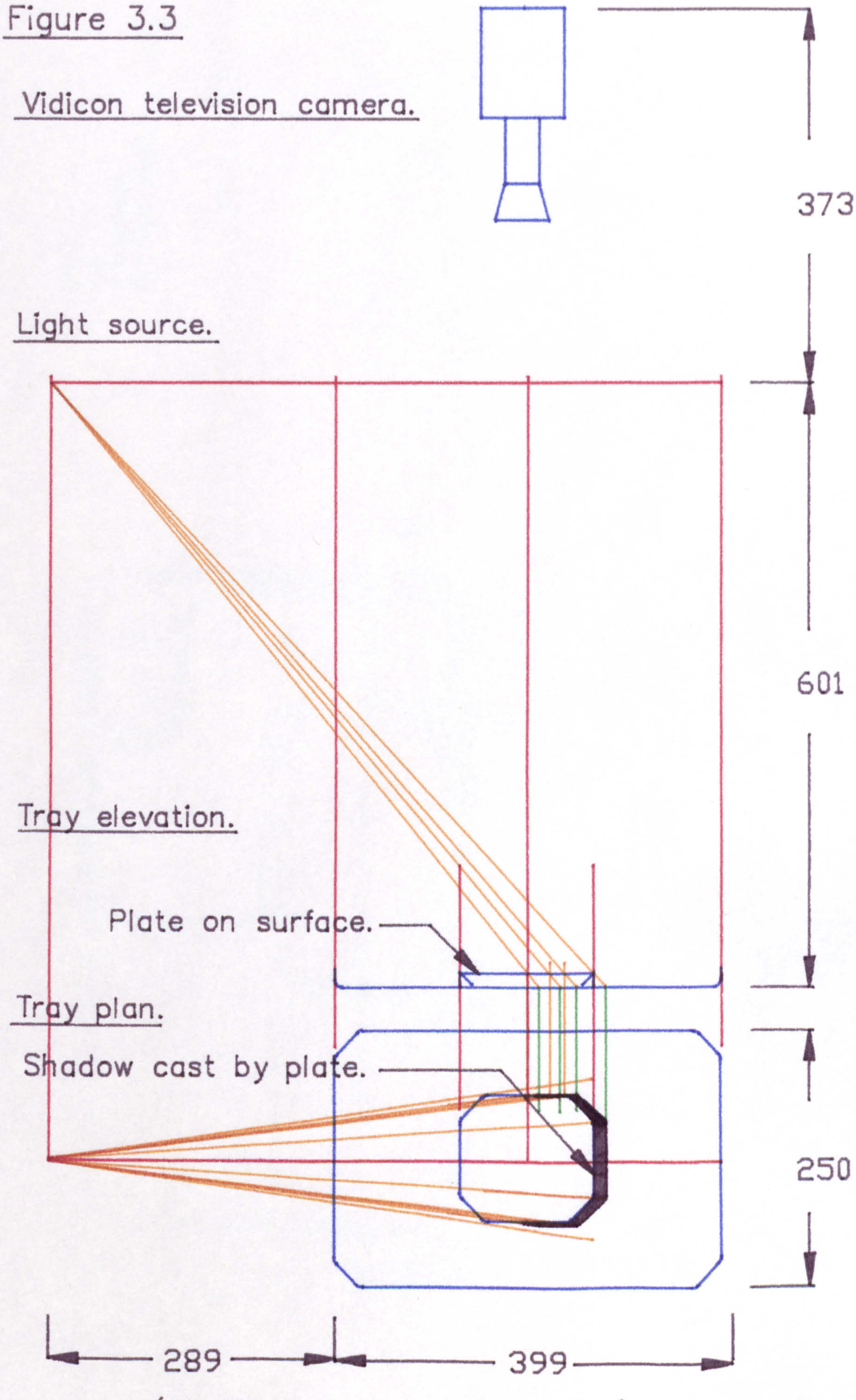
SCHEMATIC DIAGRAM OF ROTARY TABLE.

Figure 3.2a

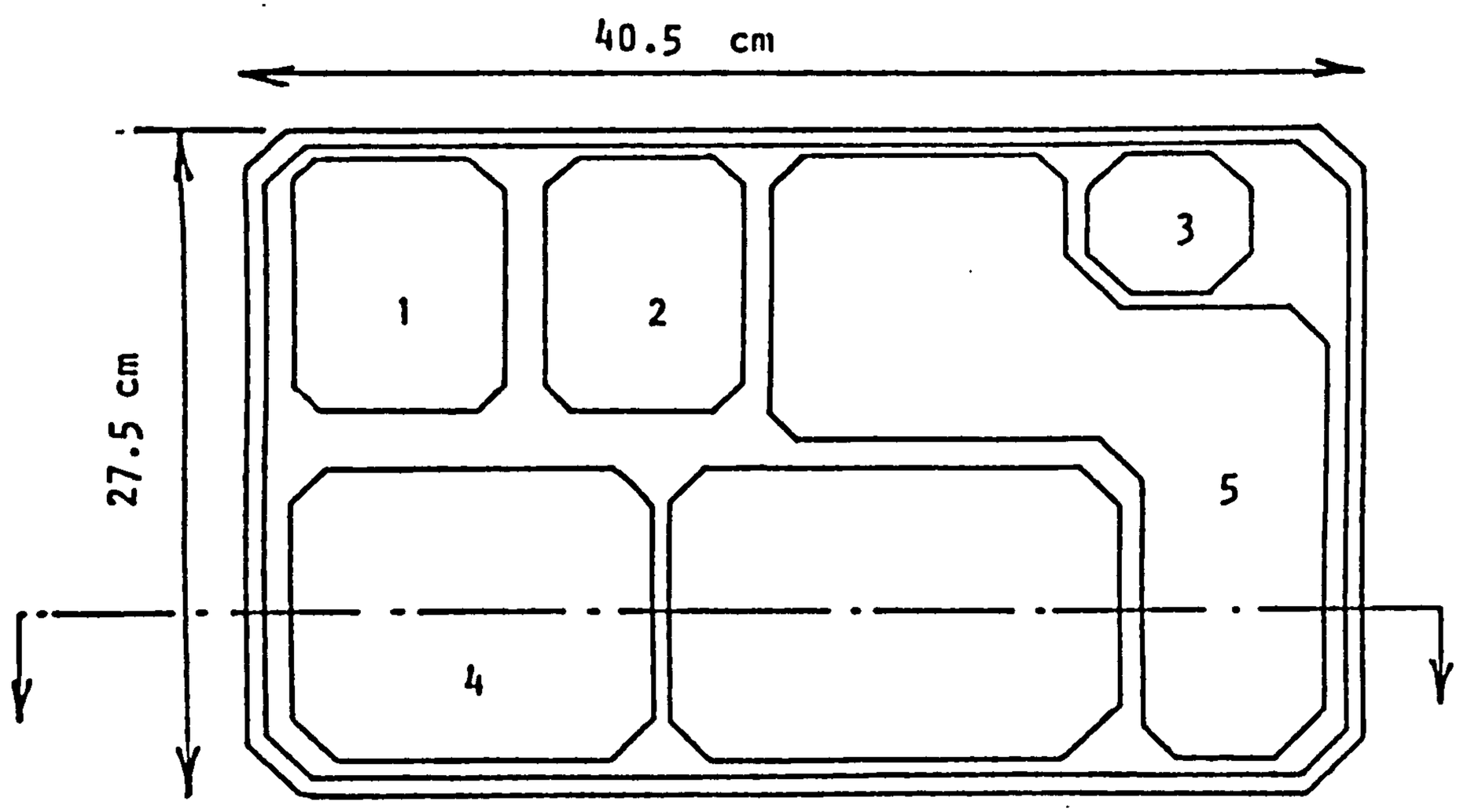


Scale Drawing of Tray/Light Source Arrangement.

Figure 3.3



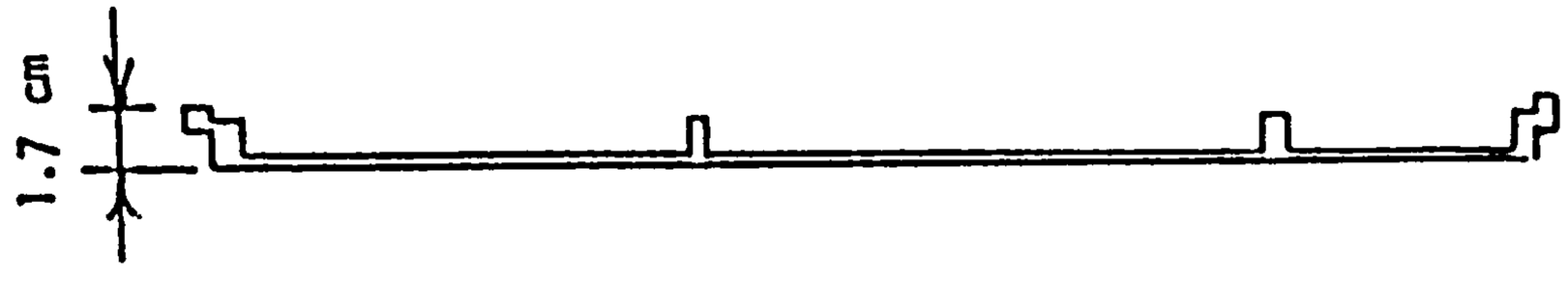
(All dimensions in millimeters.)



KEY

1,2	Bowls
3	Cup
4	Sideplate
5	Cutlery Well

Figure 3.4



SCHEMATIC DIAGRAM OF TRAY.

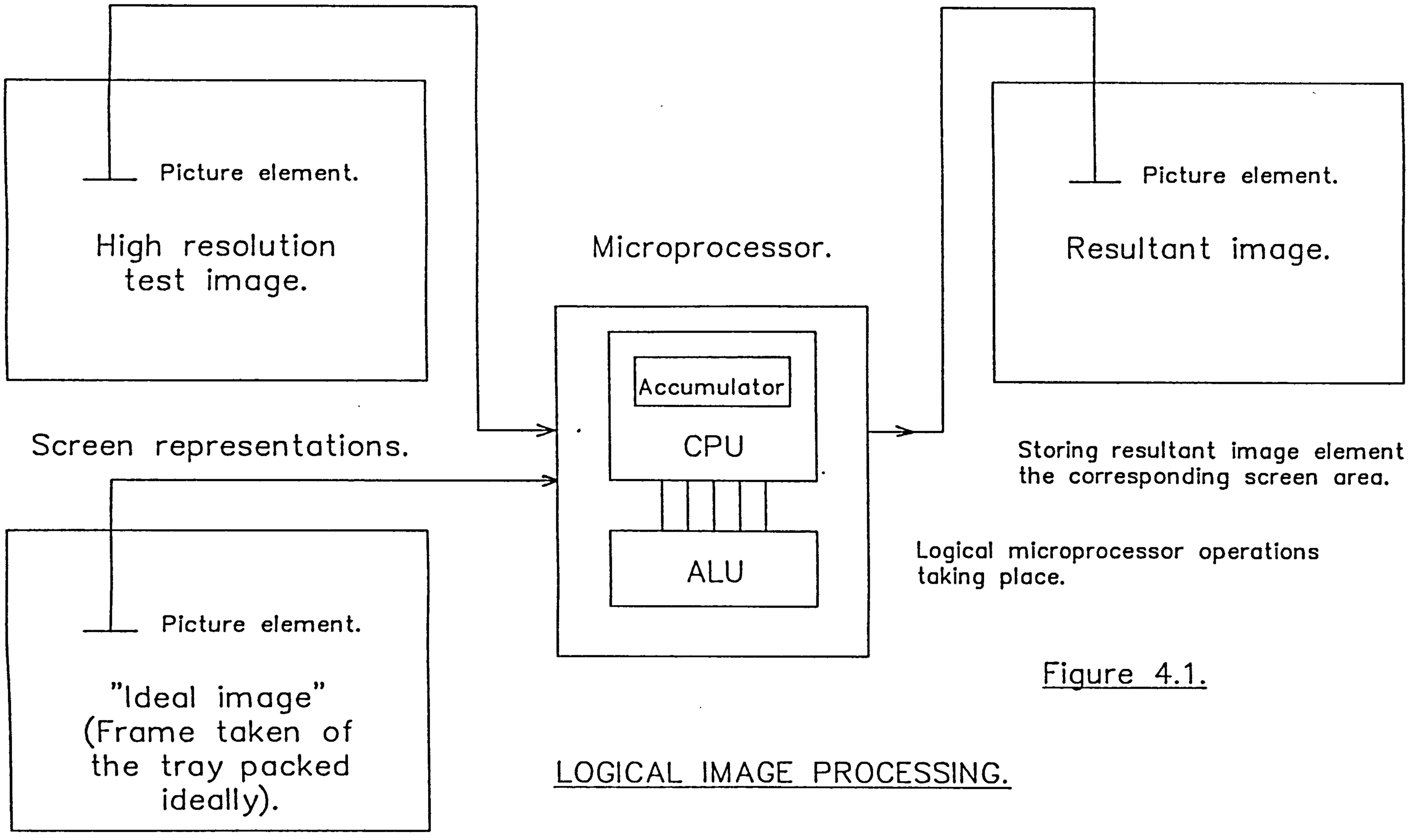


Figure 4.1.

LOGICAL IMAGE PROCESSING.

TRAY INSPECTION SEQUENCE FOR PROJECT ONE.

Figure 4.2.

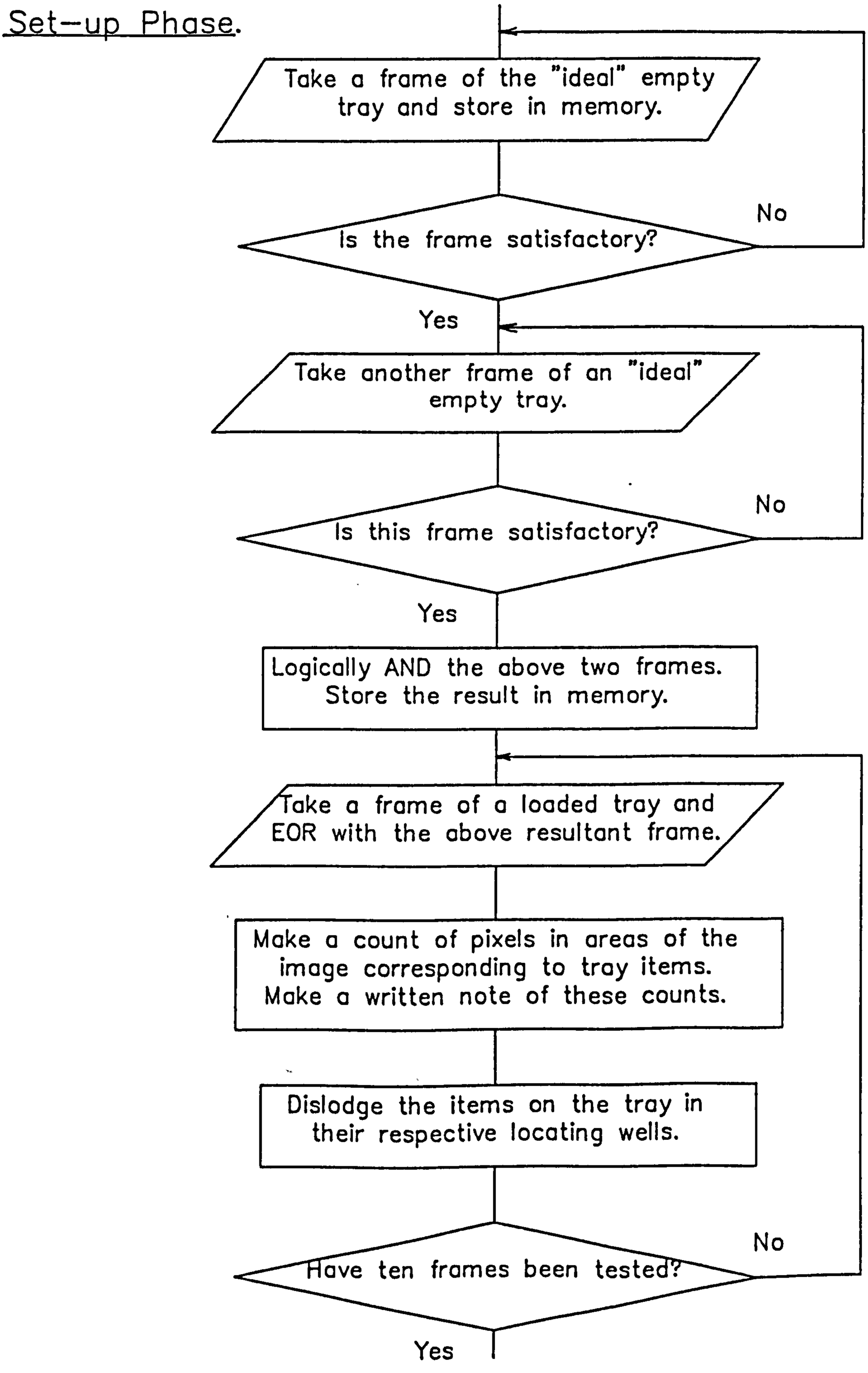
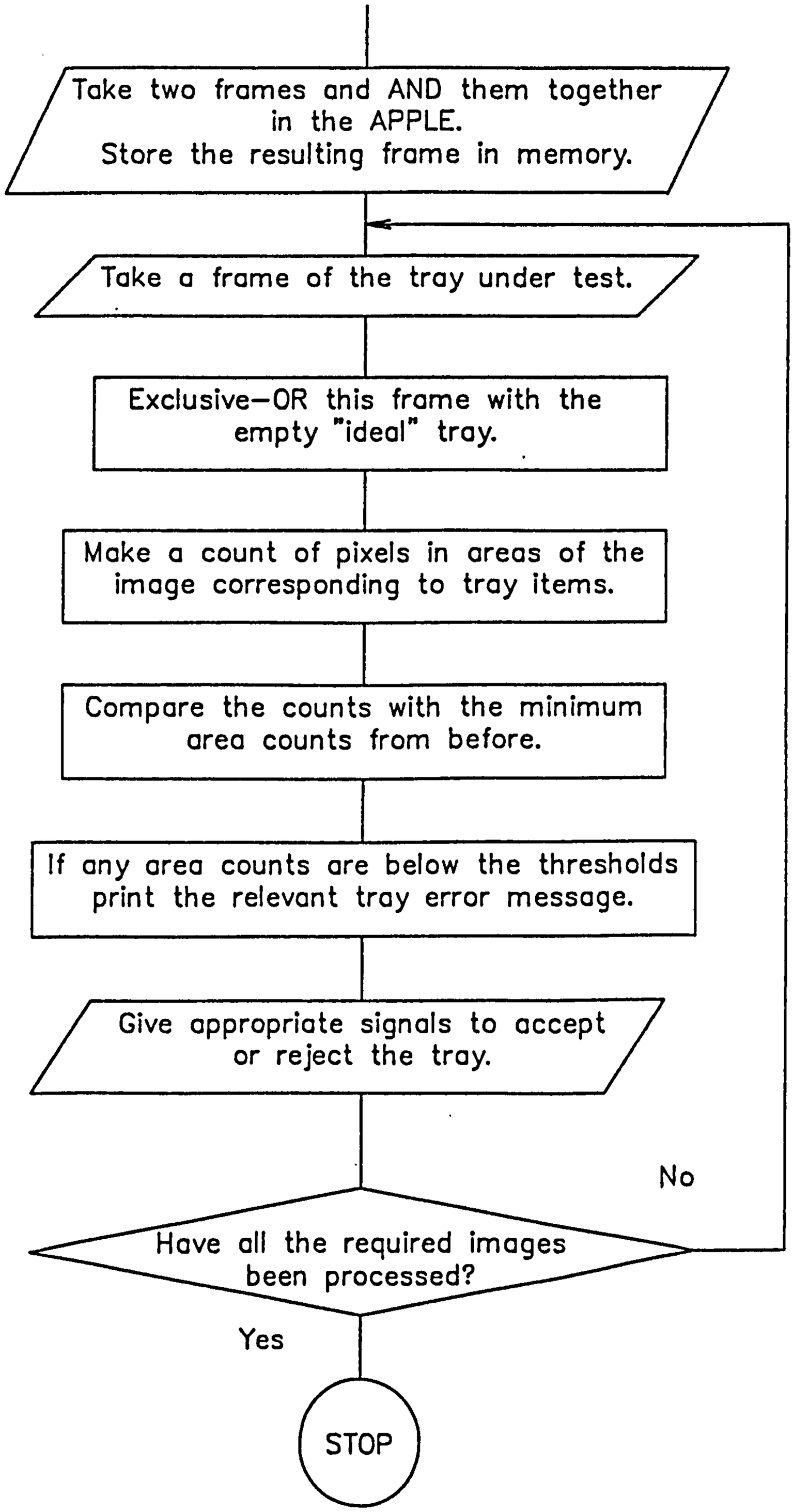


Figure 4.2 continued.

Tray Testing Phase.



Flow Diagram of the First Project DUMC Unit Development Software.

Figure 4.3

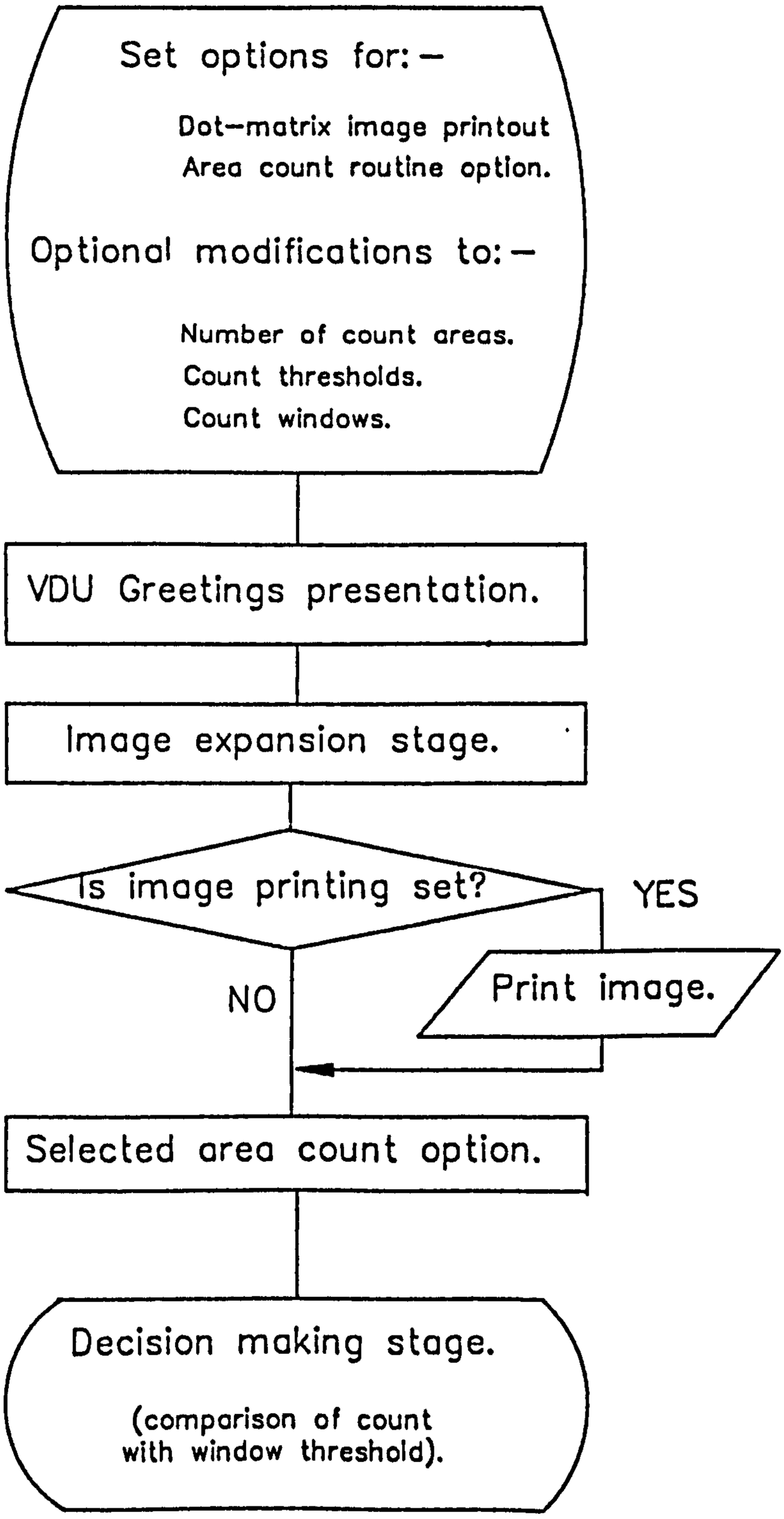


Figure 5.1

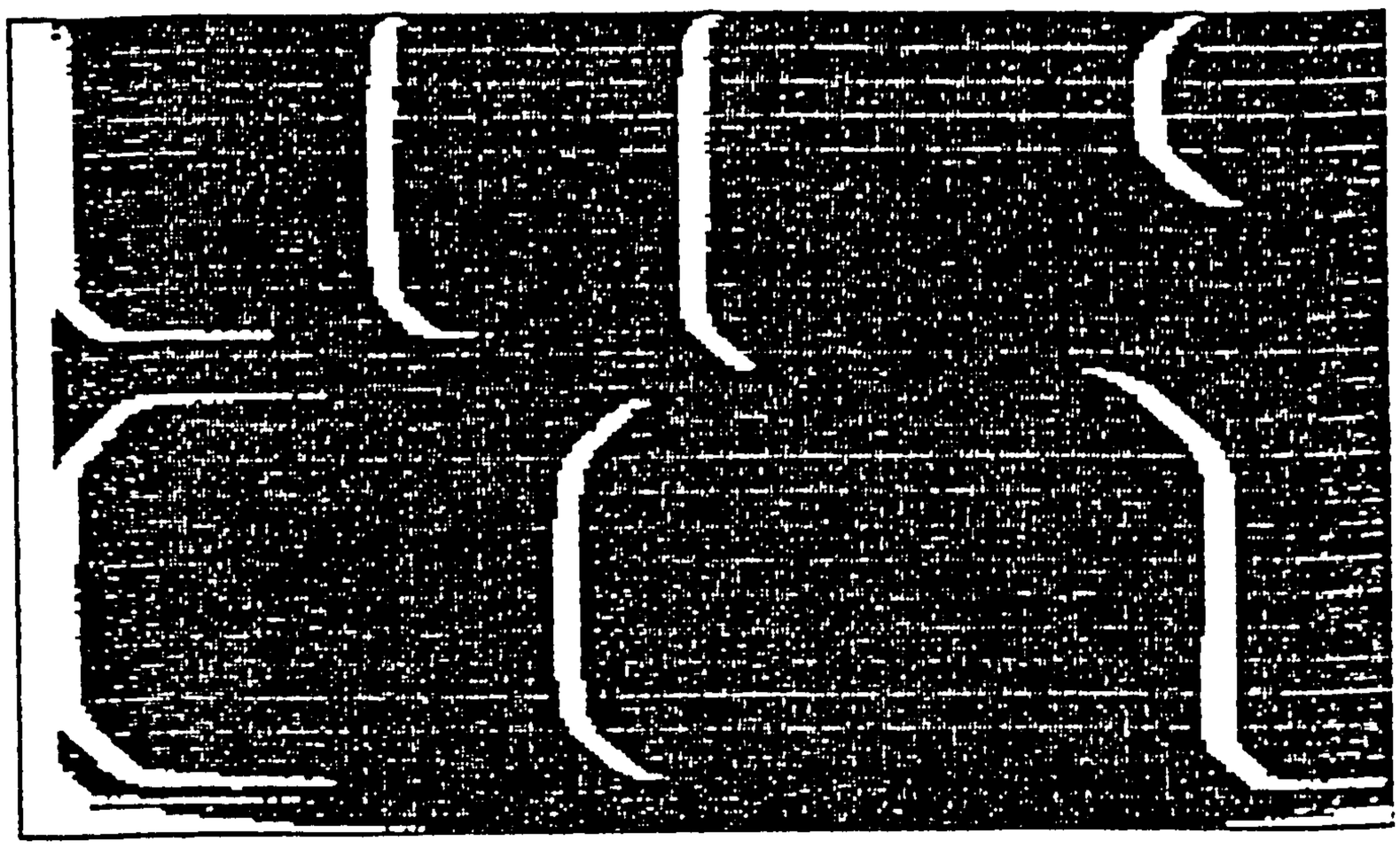


Figure 5.2

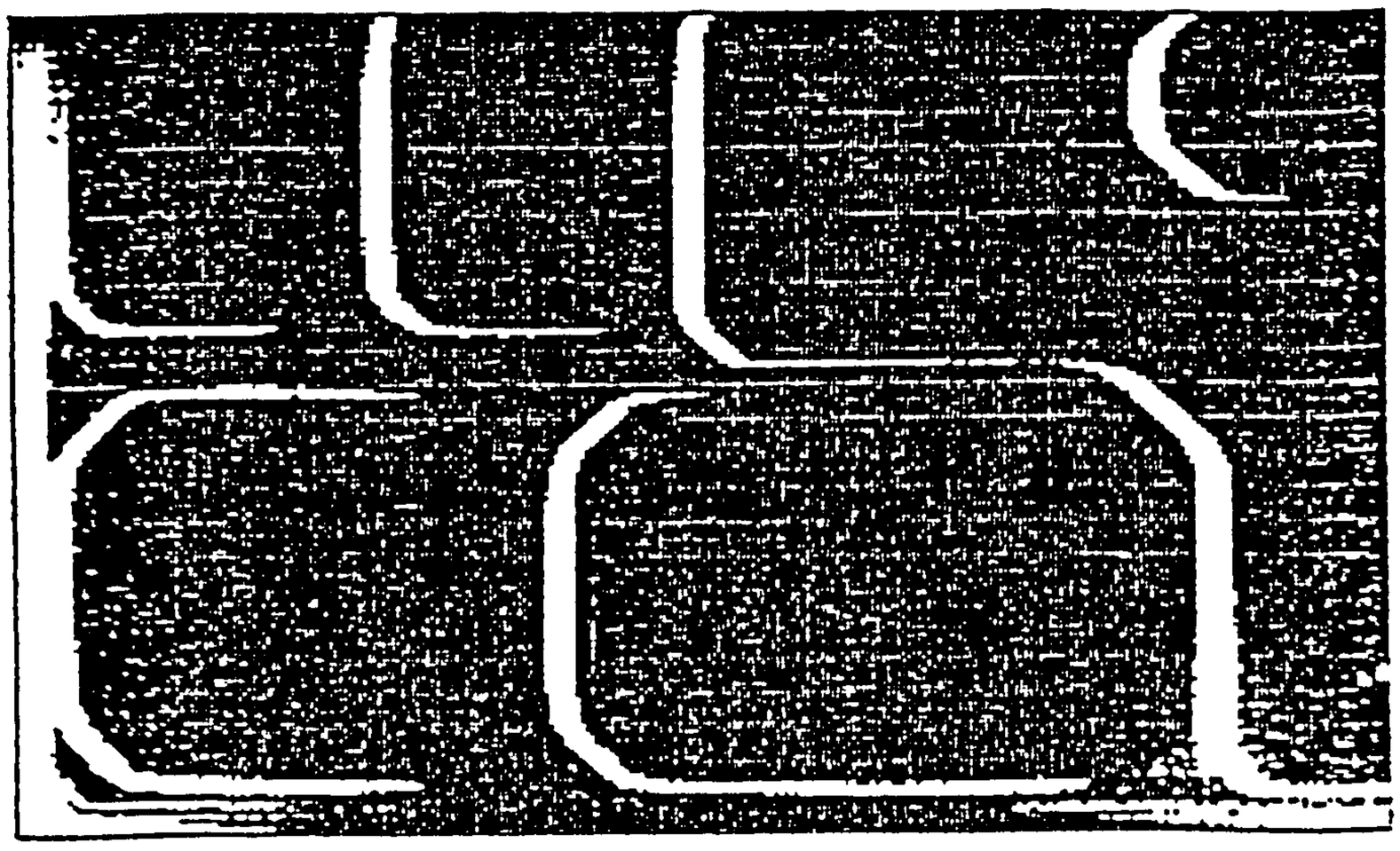


Figure 5.3

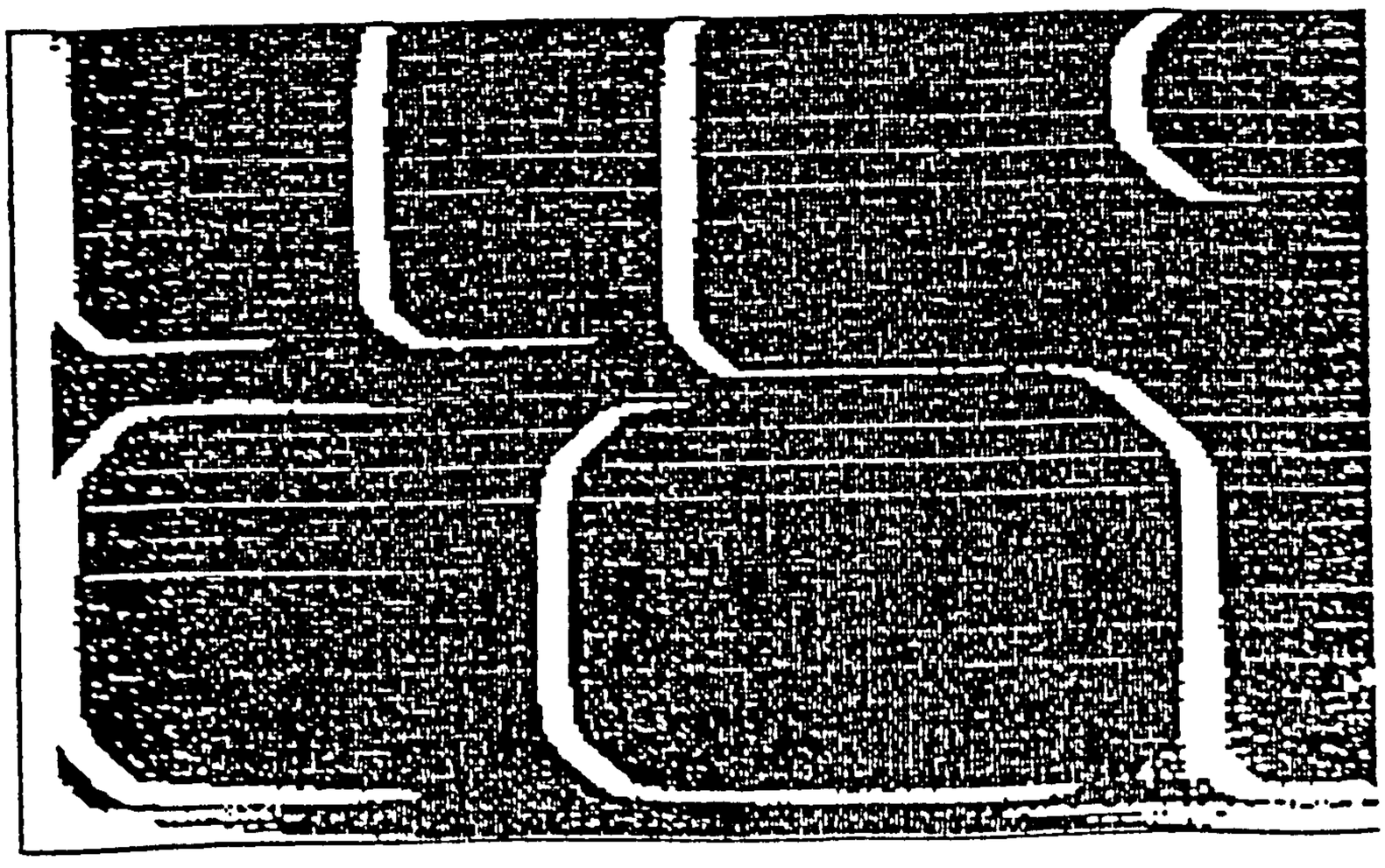


Figure 5.4

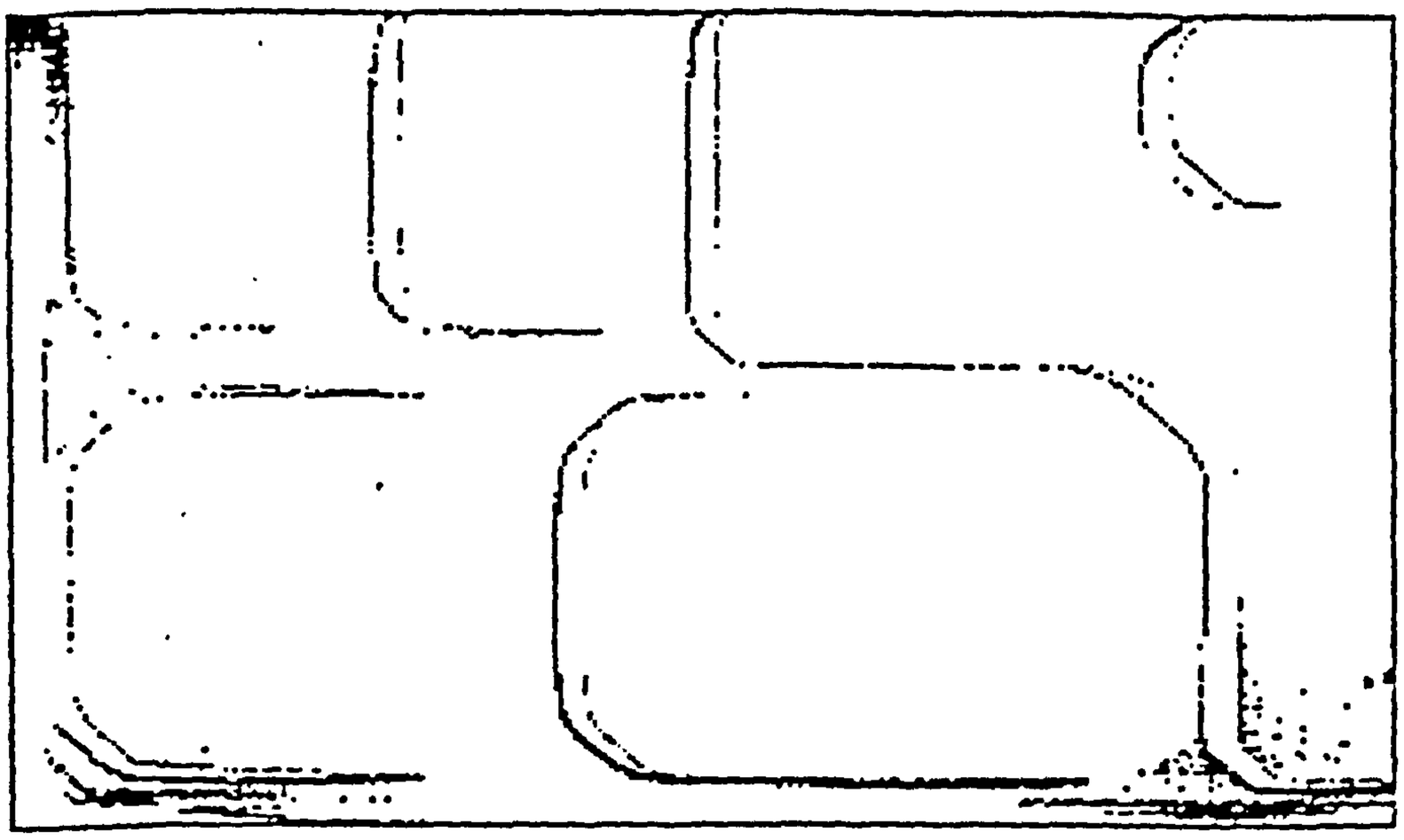


Figure 5.5

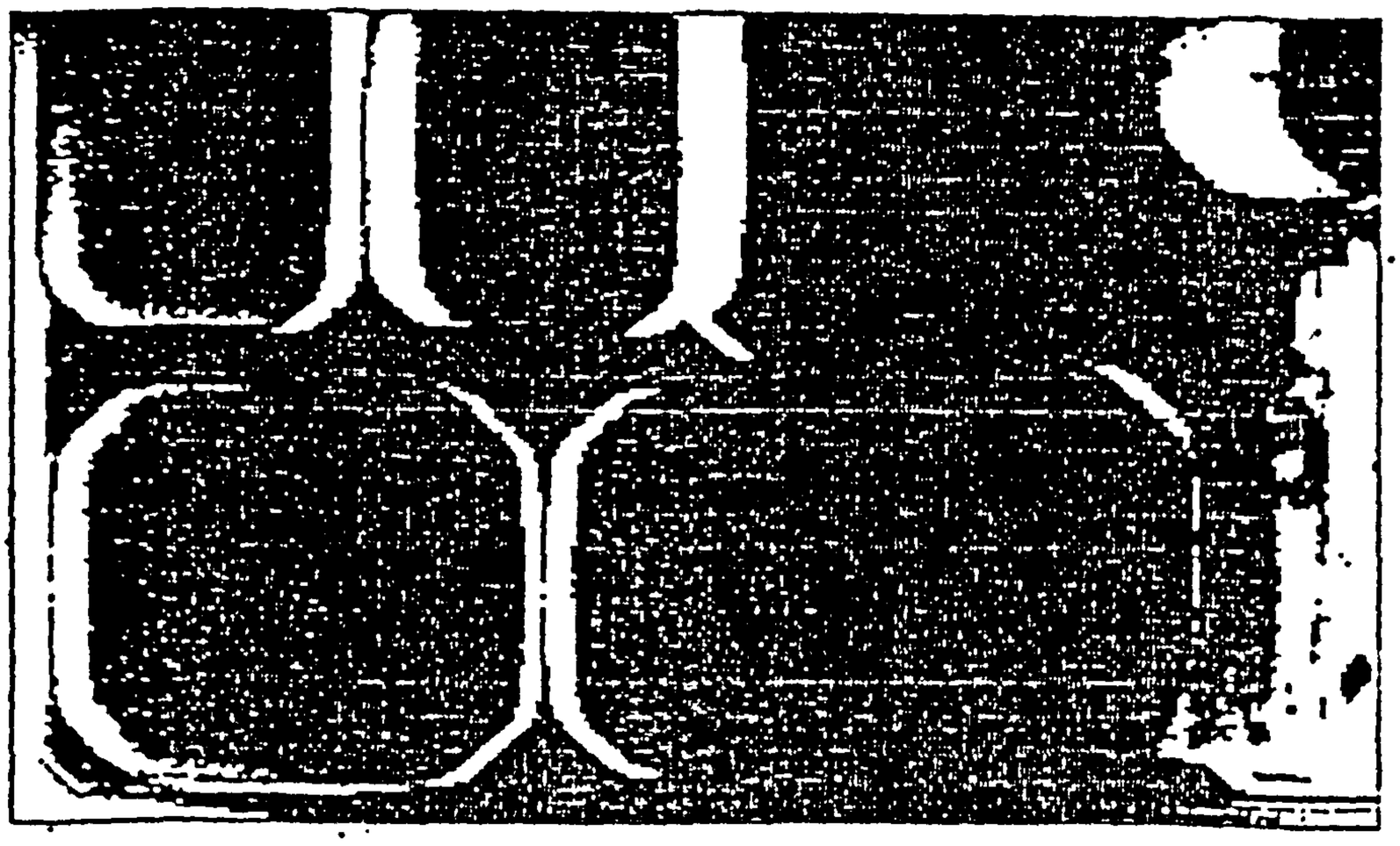


Figure 5.6

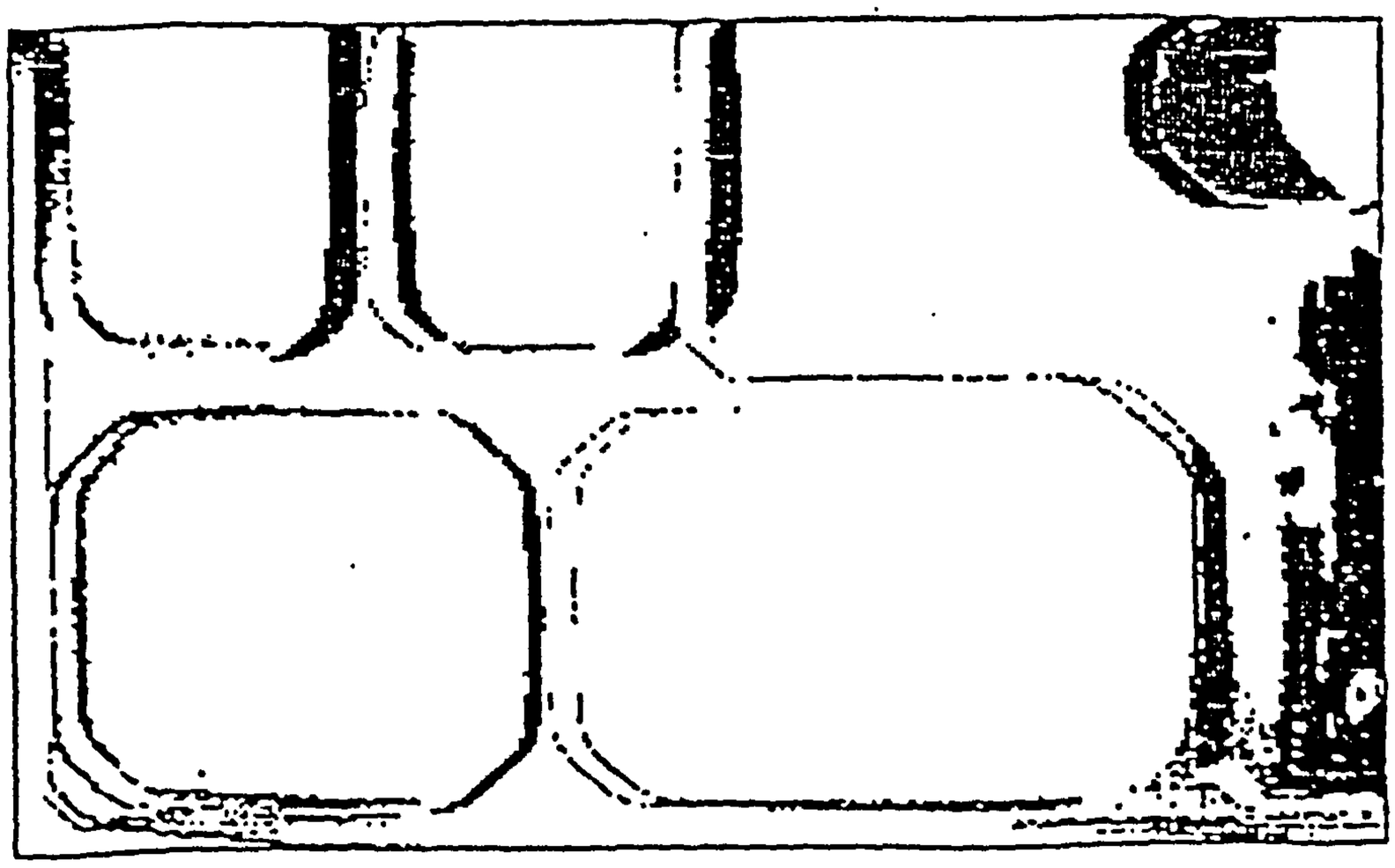
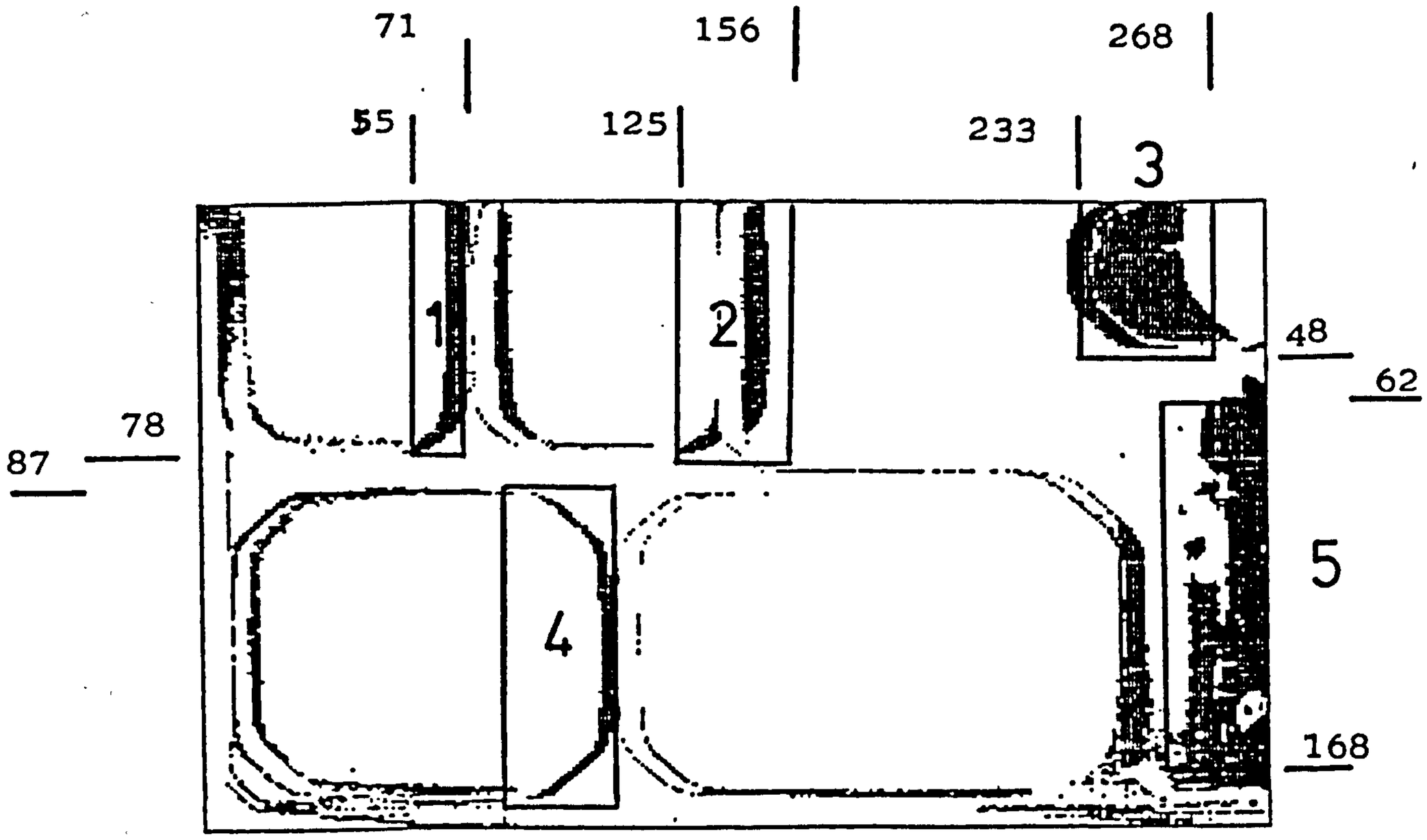


Figure 5.7

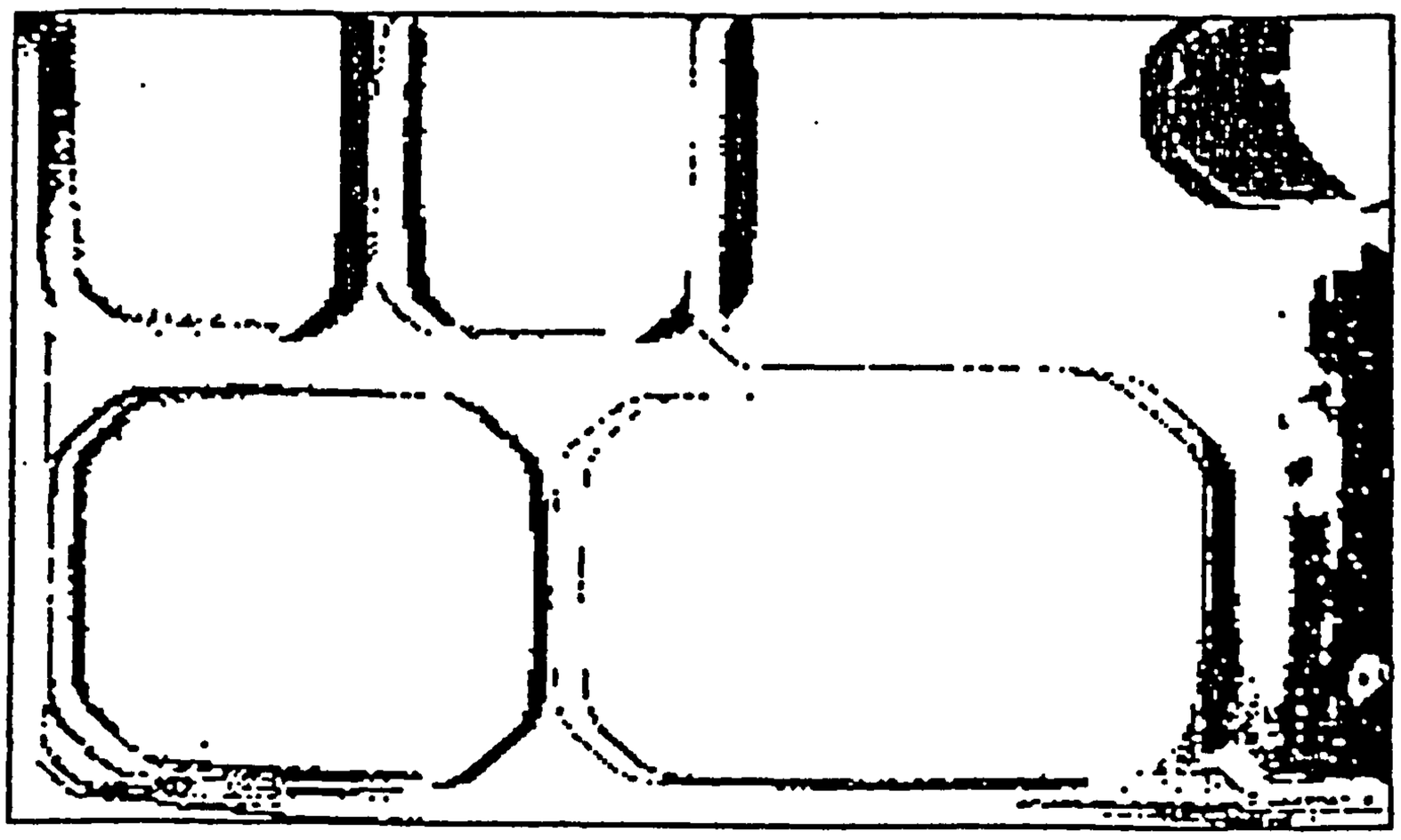


Print of current image being worked on.

81 | | 110 253 |

FIGURE 5.8

```
**************************************************
*                                                *
*            68000  INSPECTION  SOFTWARE         *
*                                                *
*  For DURHAM UNIVERSITY ENGINEERING DEPARTMENT  *
*                                                *
*             BRITISH  AIRWAYS  PROJECT          *
*                                                *
*                    Created  September 1984     *
*                                                *
**************************************************
transfer into bits complete
```



```
 Print of current image being worked on.

          Counting in 5 areas.

Your two upper left point co-ords are:  55        0
Your two bottom right co-ords are:      71        78

Your two upper left point co-ords are:  125       0
Your two bottom right co-ords are:      156       78

Your two upper left point co-ords are:  233       0
Your two bottom right co-ords are:      268       48

Your two upper left point co-ords are:  81        87
Your two bottom right co-ords are:      110       182

Your two upper left point co-ords are:  253       62
Your two bottom right co-ords are:      279       168

     Areas corresponding are :-   488
     Areas corresponding are :-   570
     Areas corresponding are :-   1150
     Areas corresponding are :-   335
     Areas corresponding are :-   1752

Item 1 is present and located.

Item 2 is present and located.

Item 3 is present and located.

Item 4 is present and located.

Item 5 is present and located.

          Press "a" key to run program again.
```

Figure 5.9

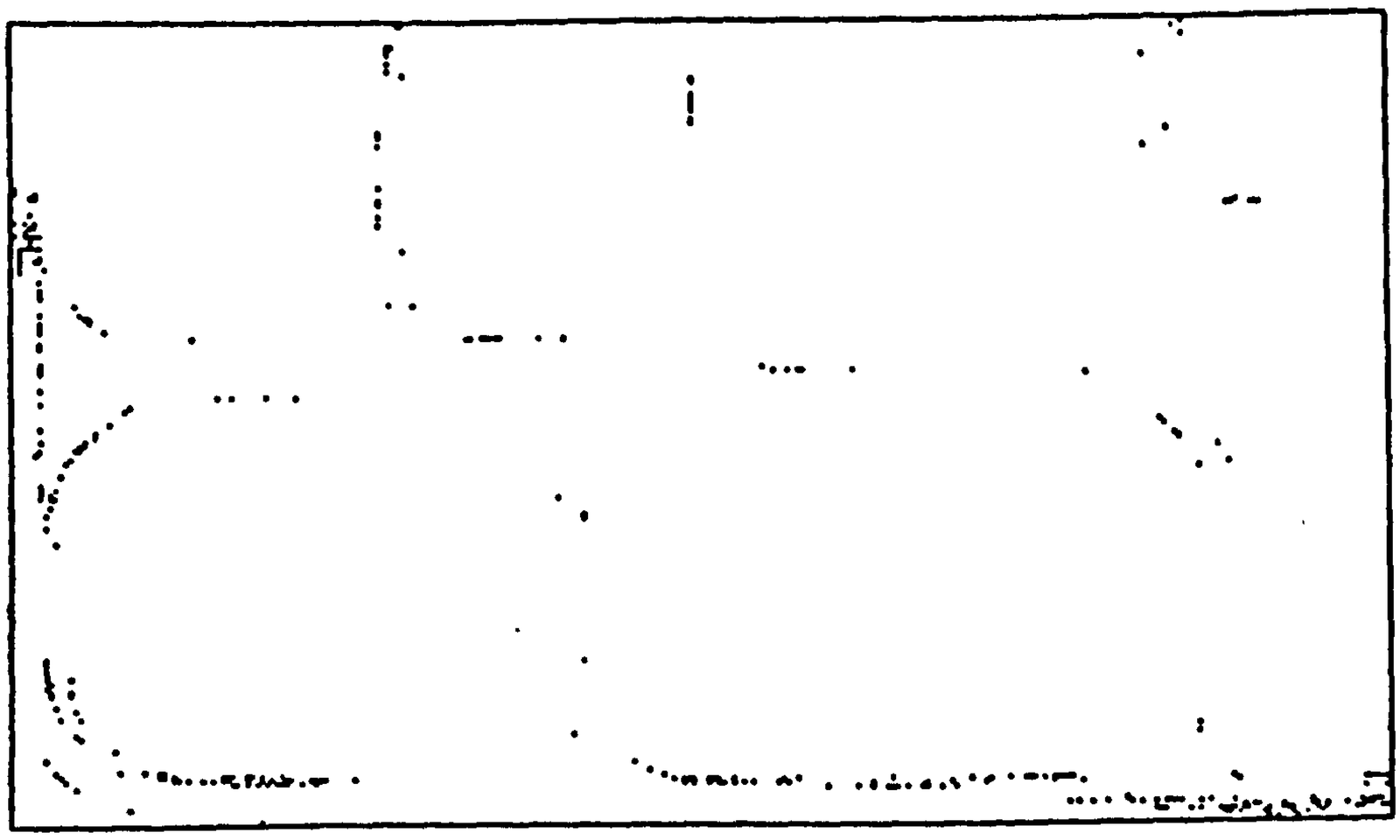


Figure 5.10

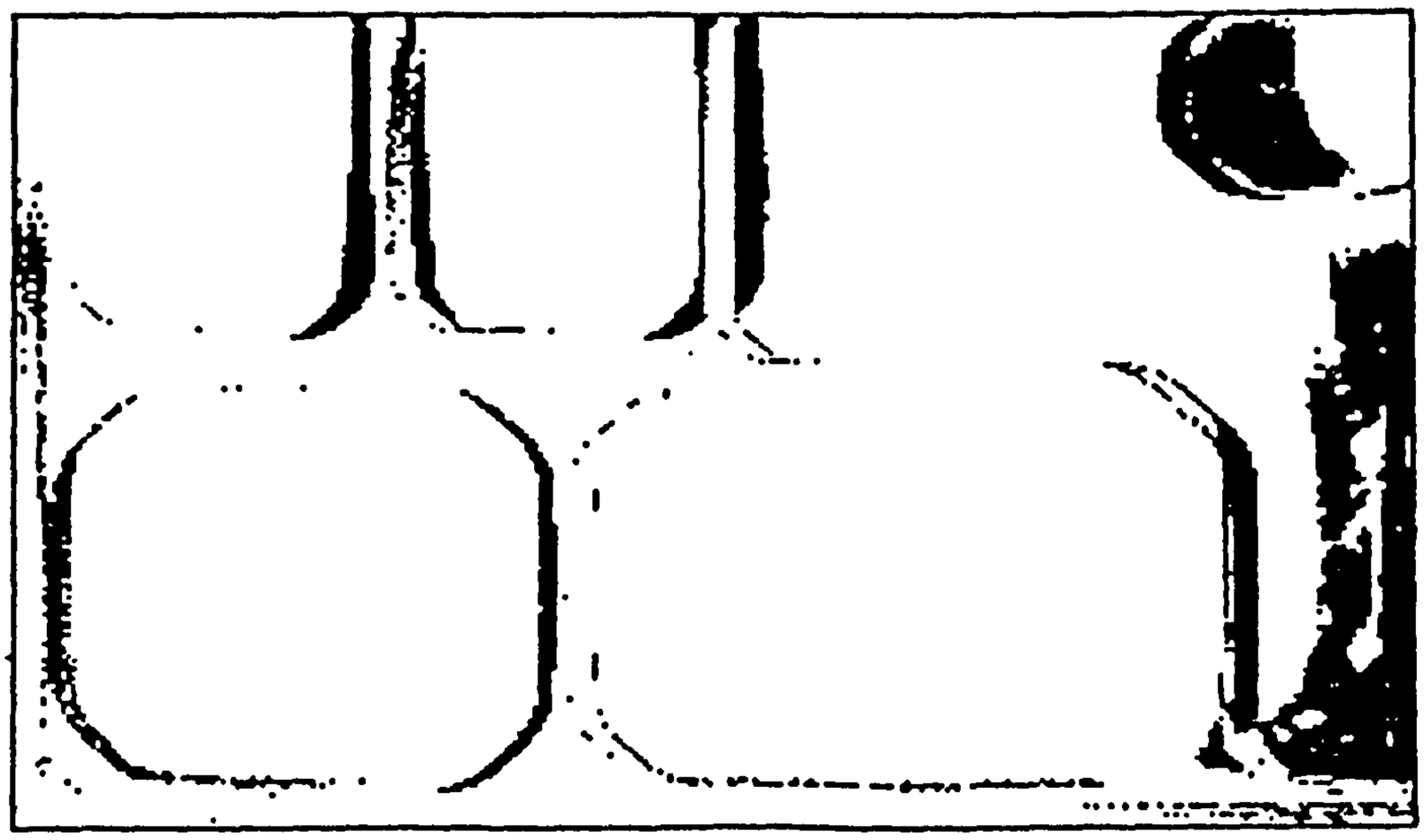


Figure 5.11

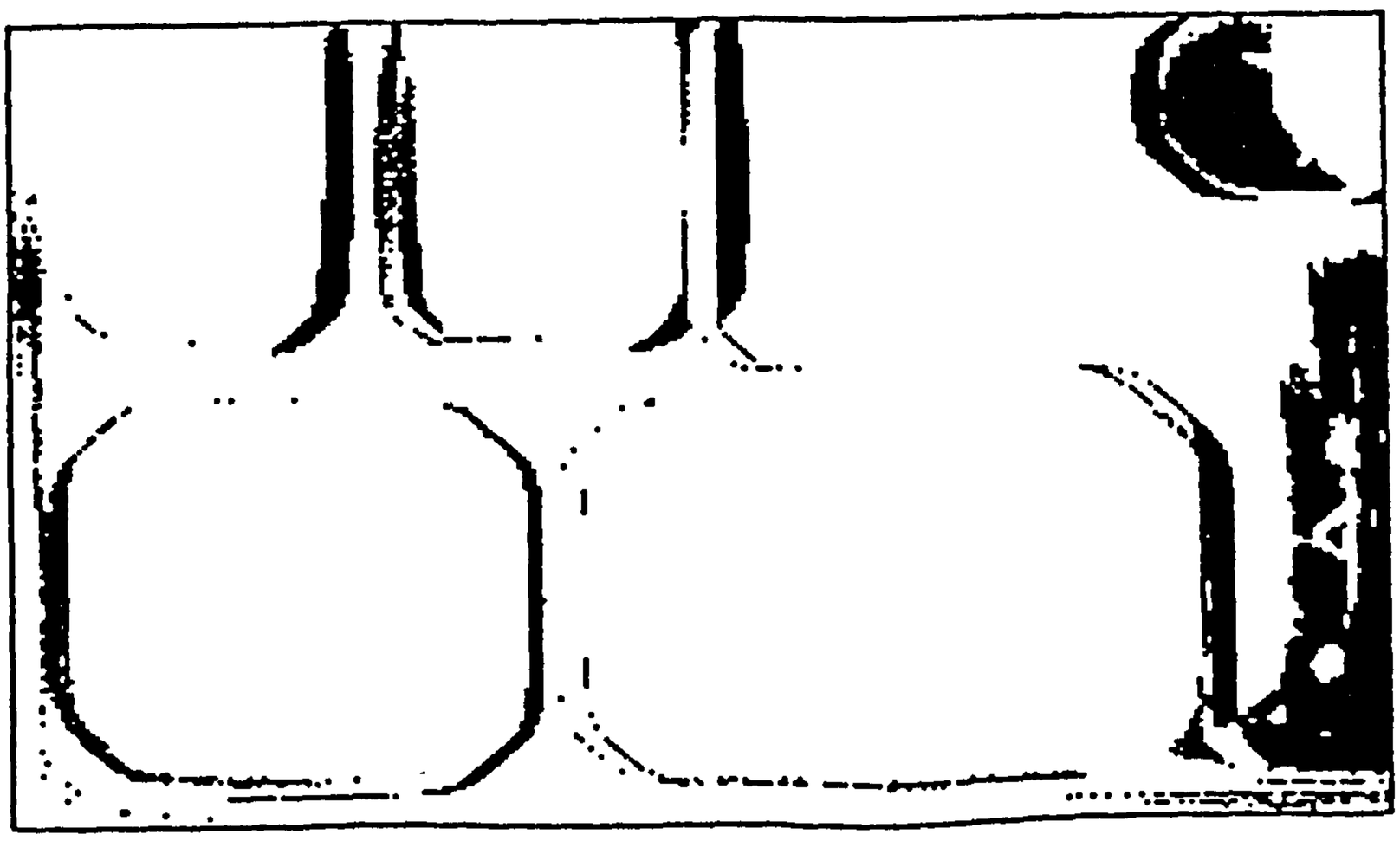


Figure 5.12

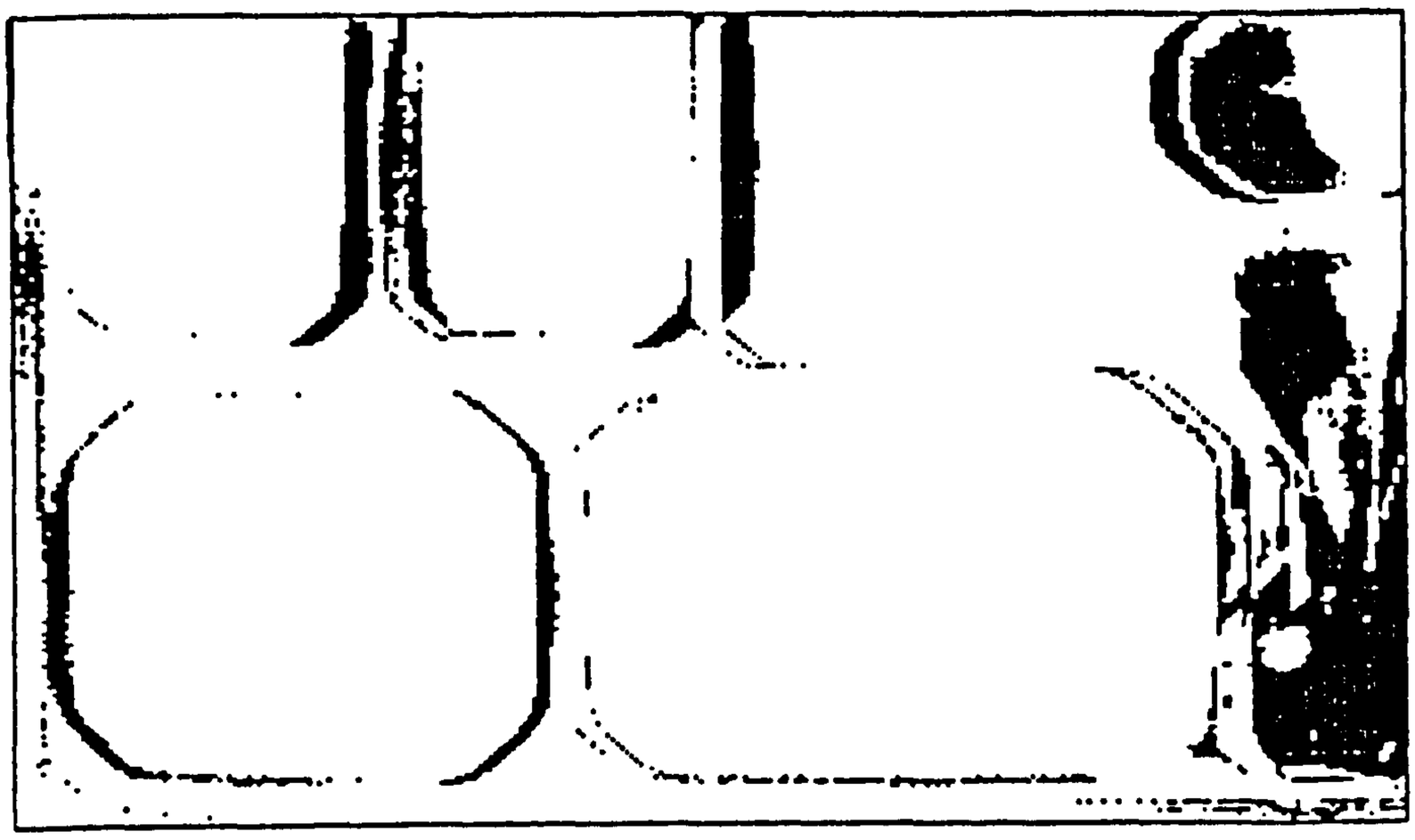


Figure 5.13

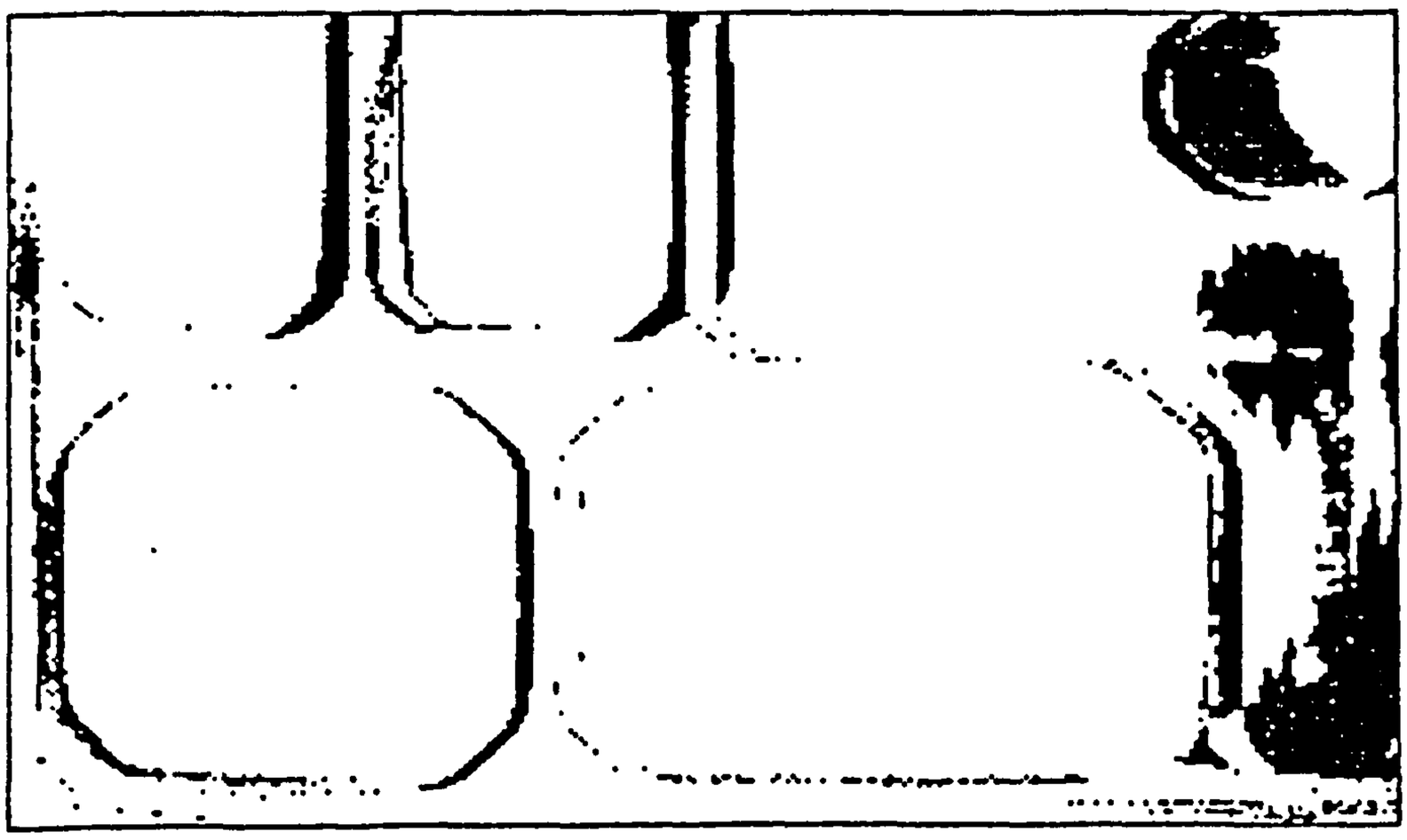


Figure 5.14

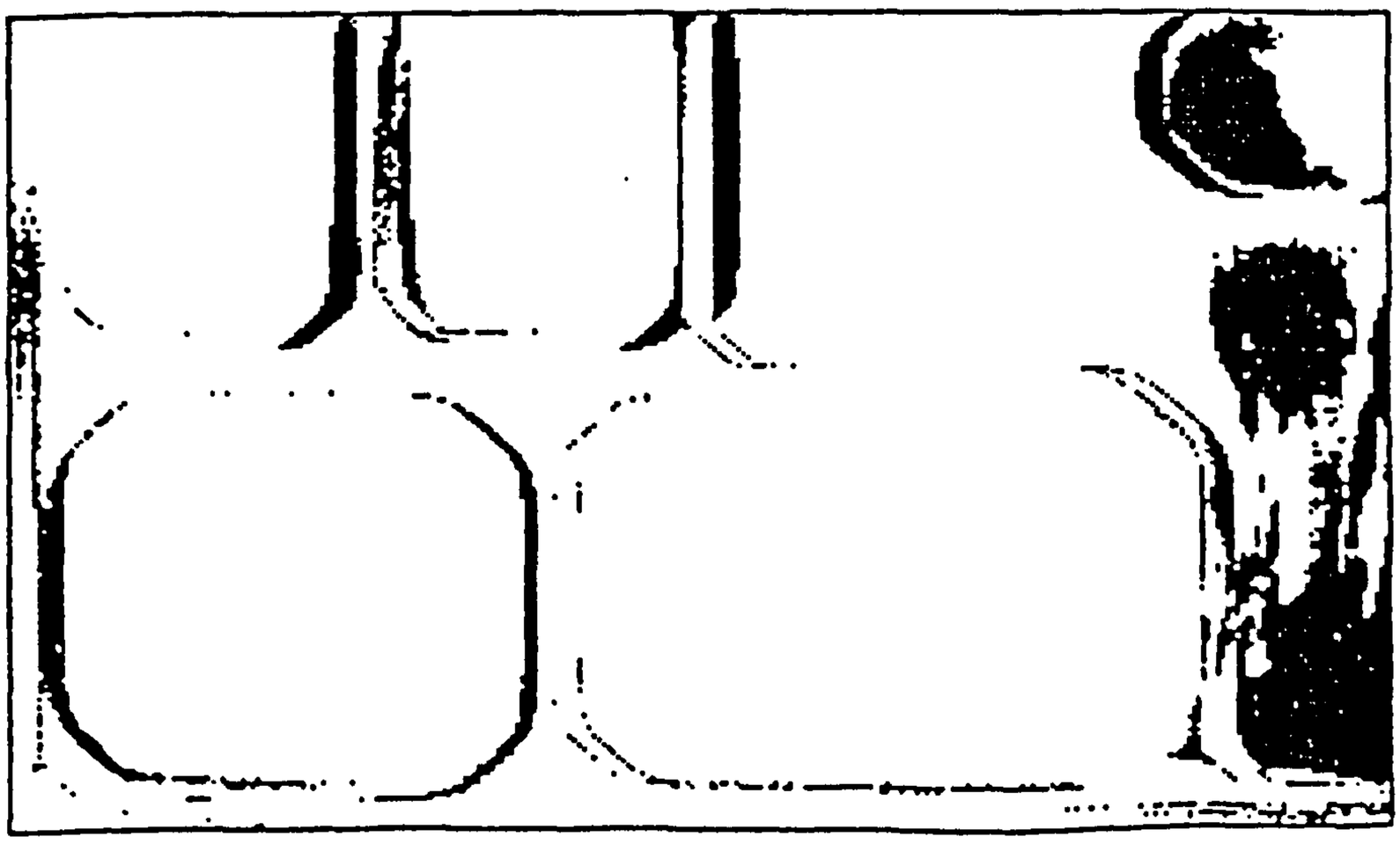


Figure 5.15

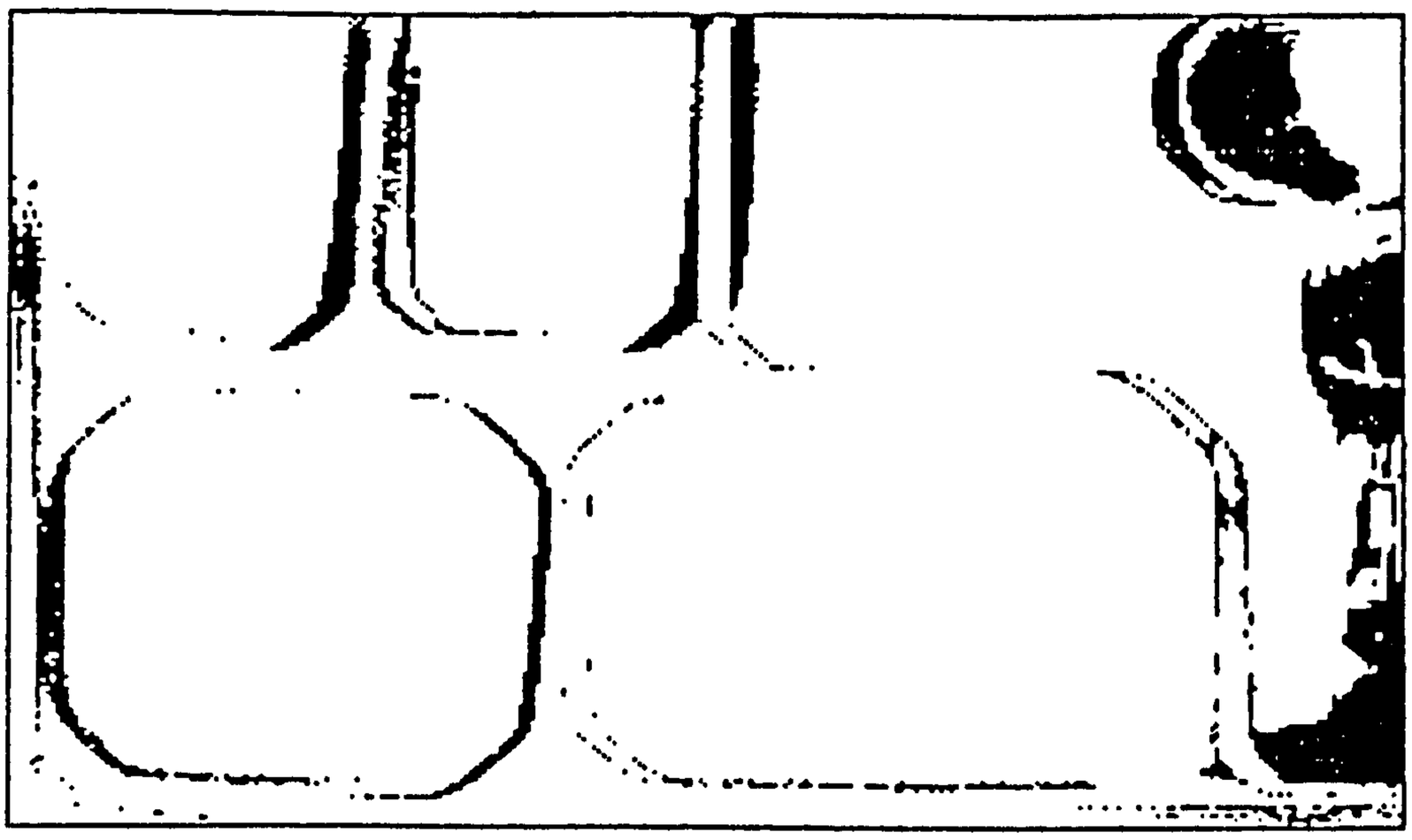


Figure 5.16

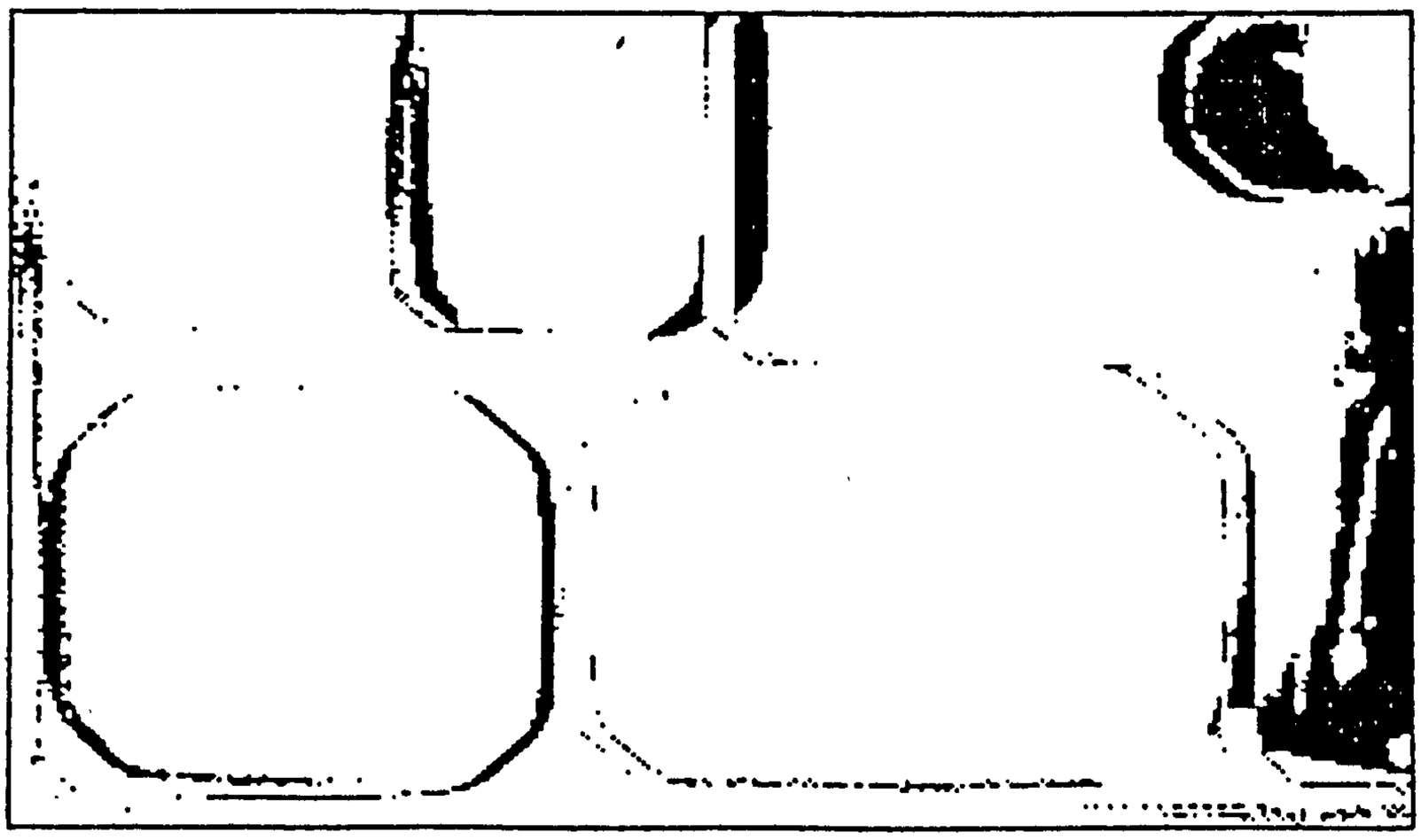


Figure 5.17

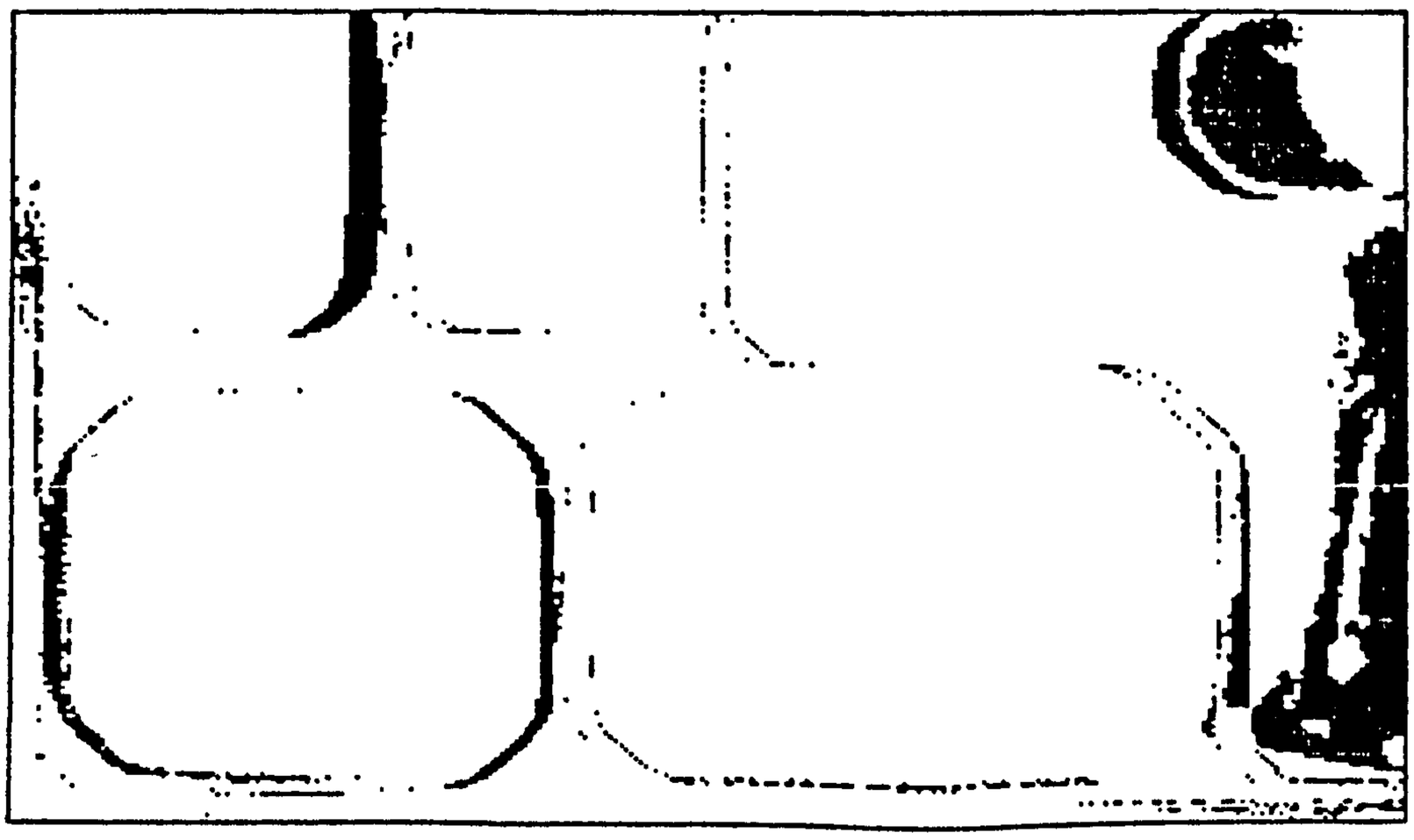


Figure 5.18

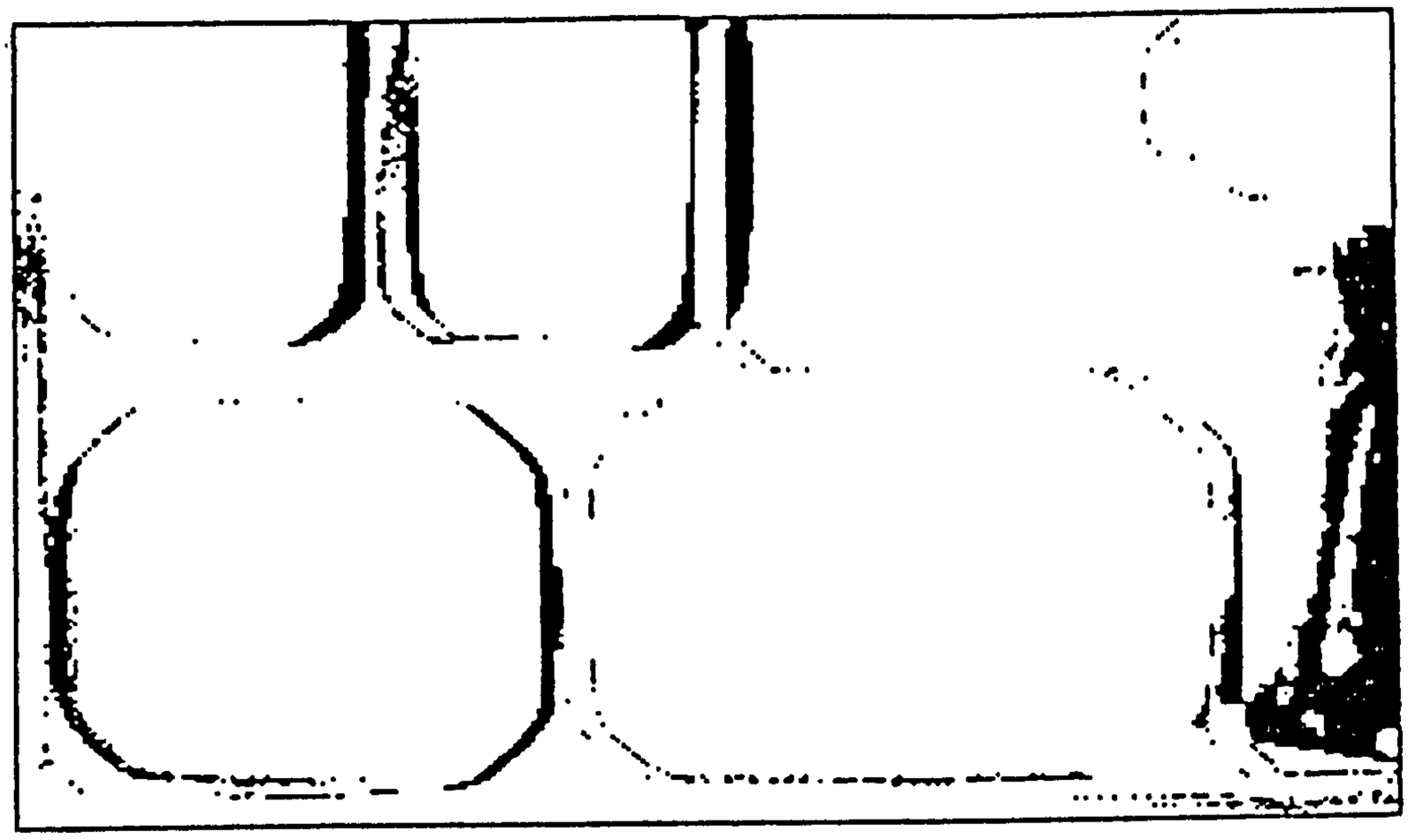


Figure 5.19

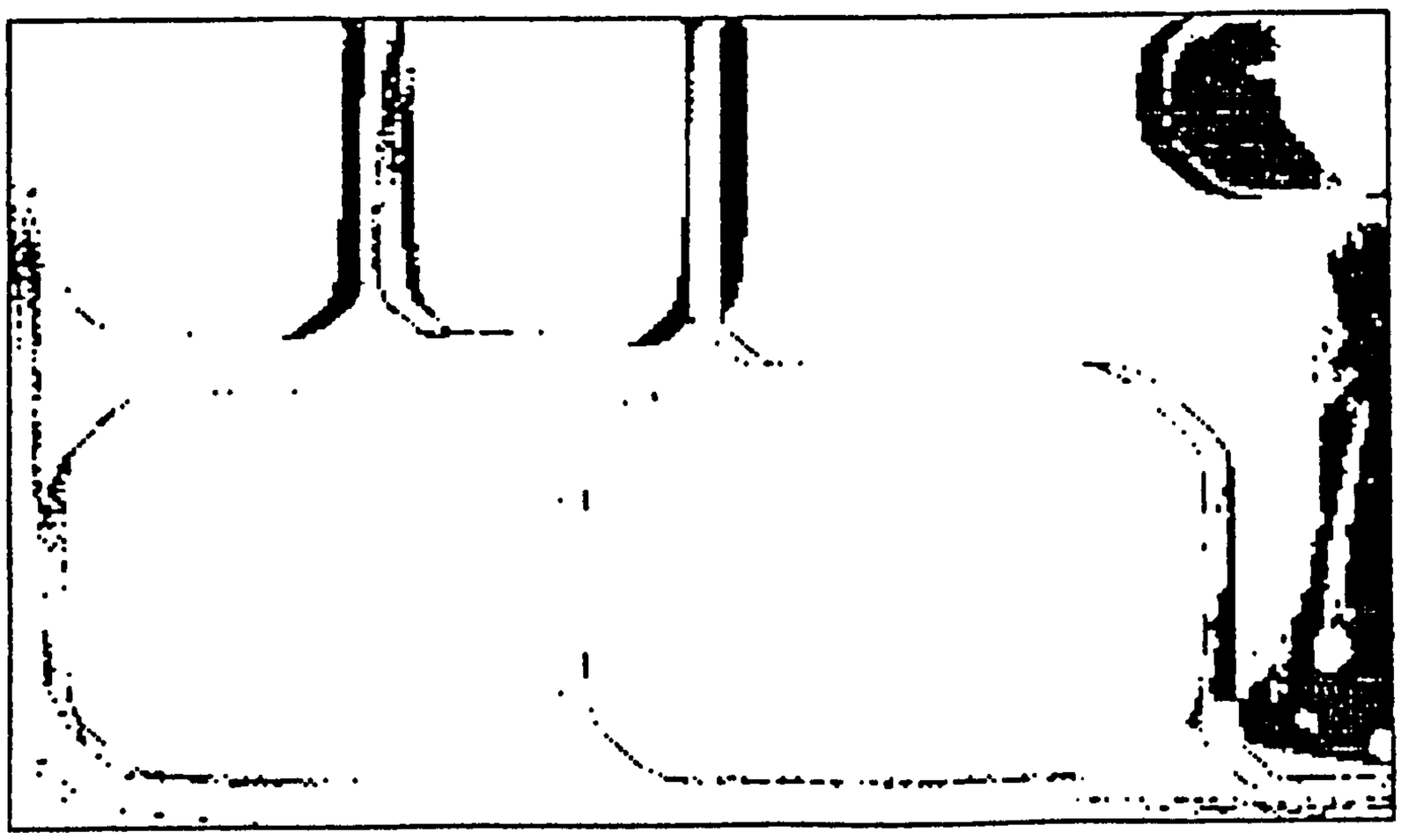


Figure 5.20

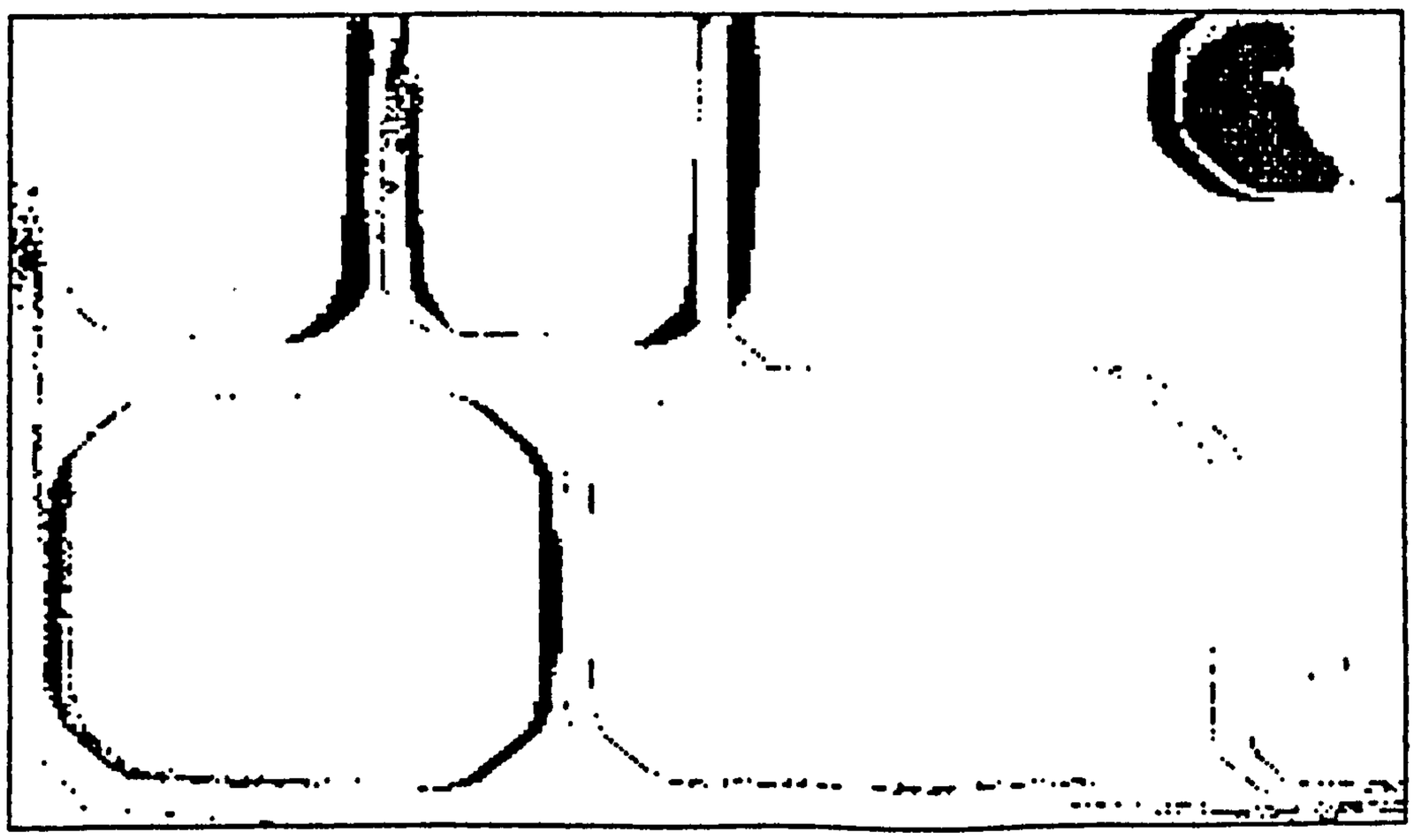


Figure 5.21

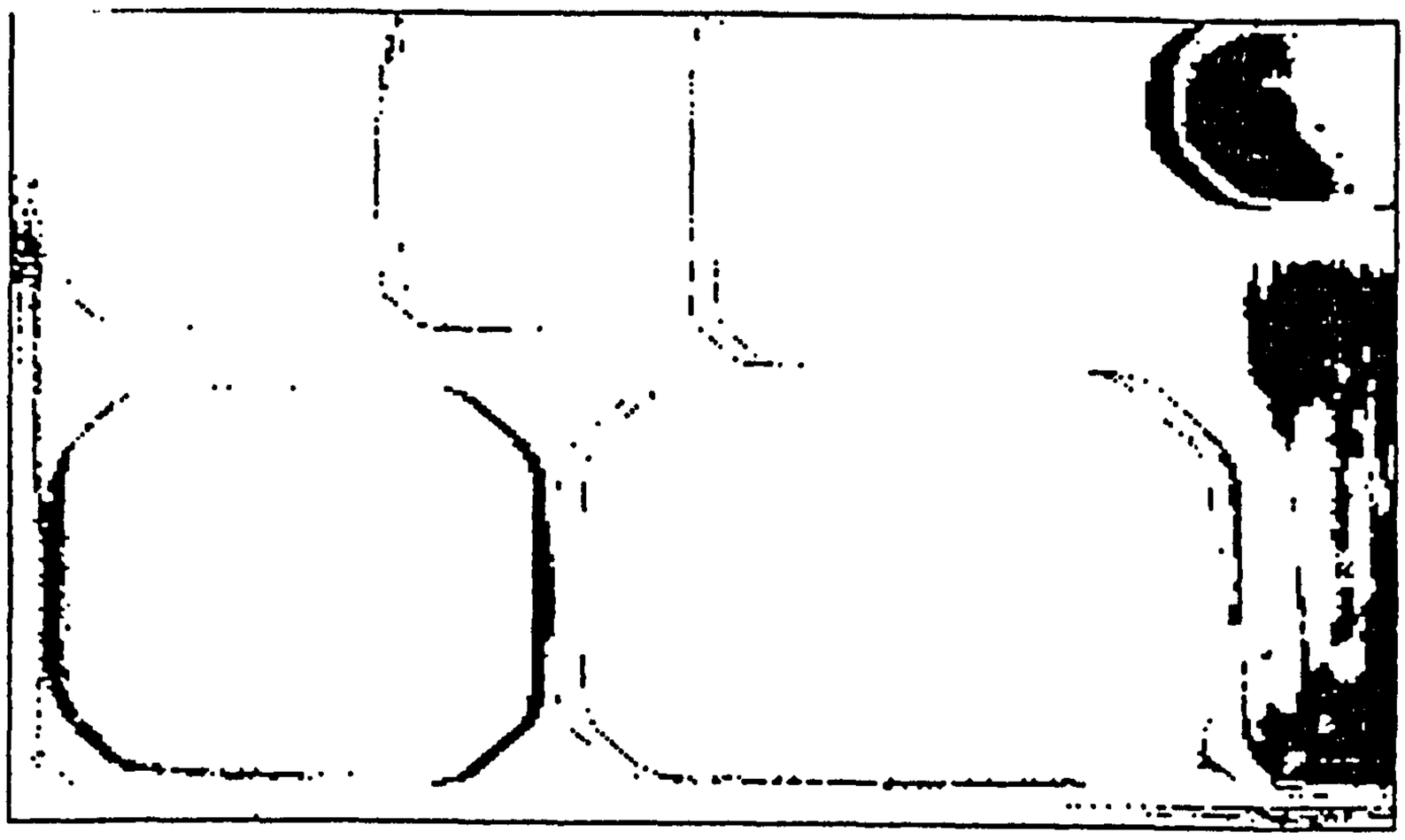


Figure 5.22

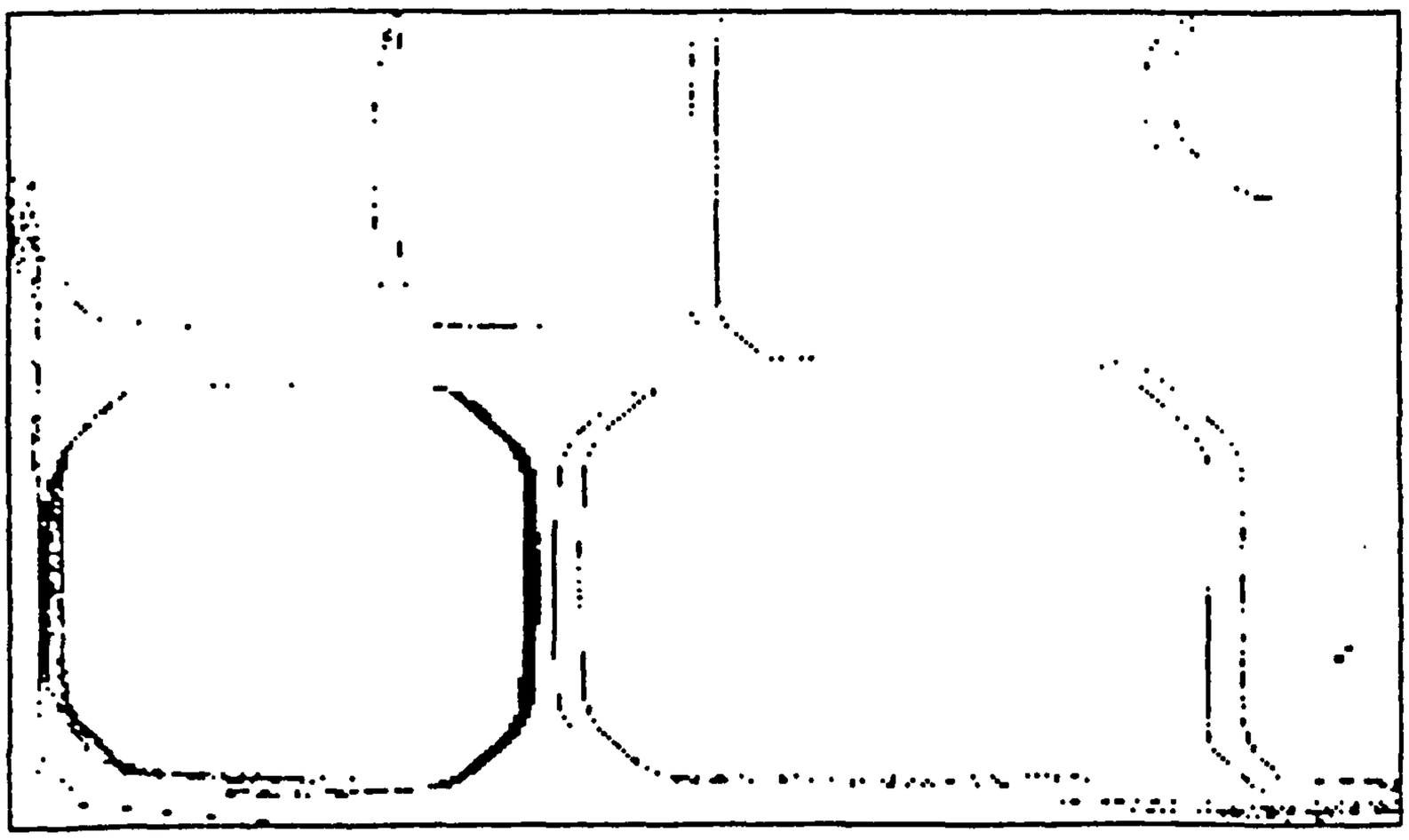


Figure 5.23a

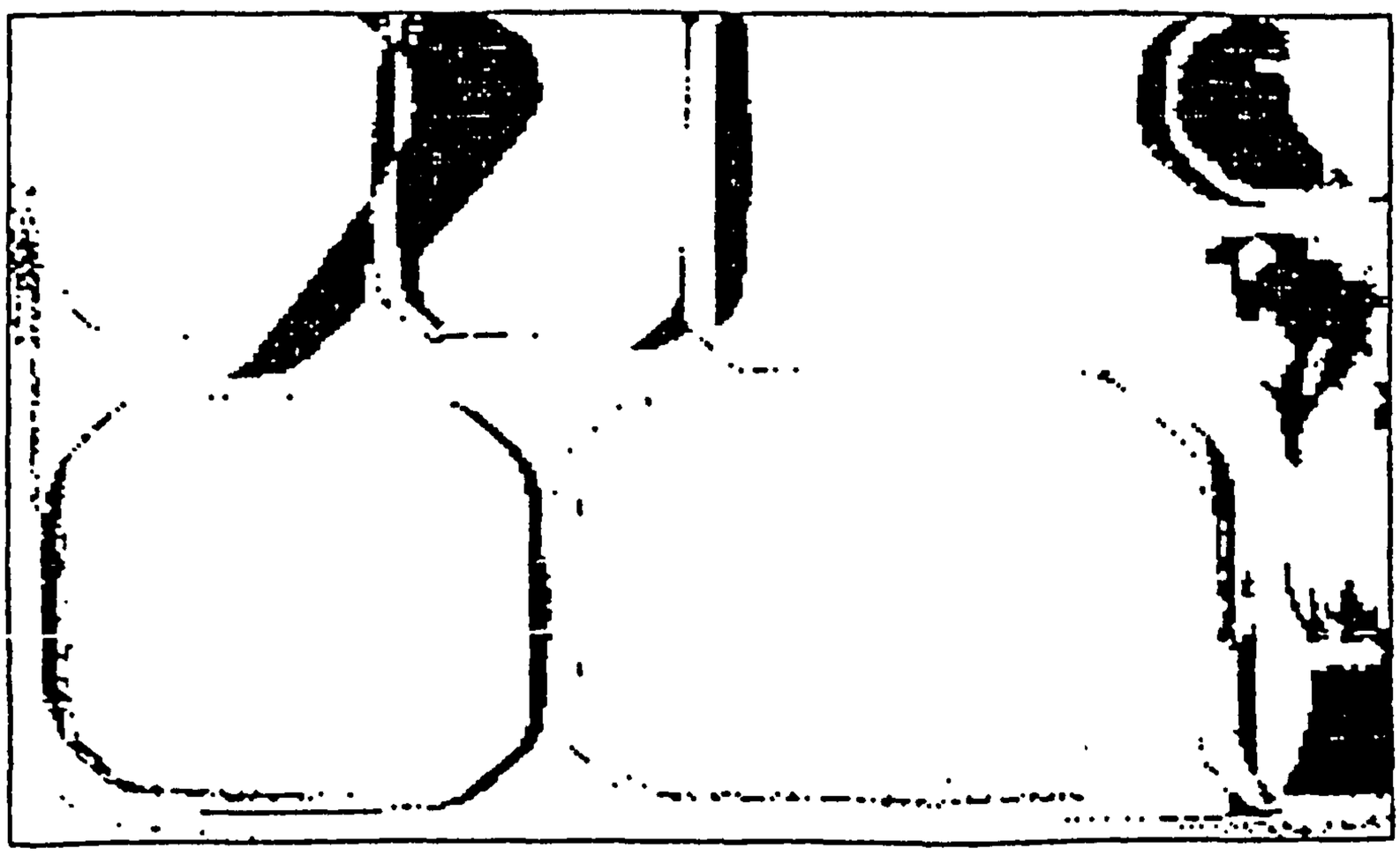


Figure 5.23b

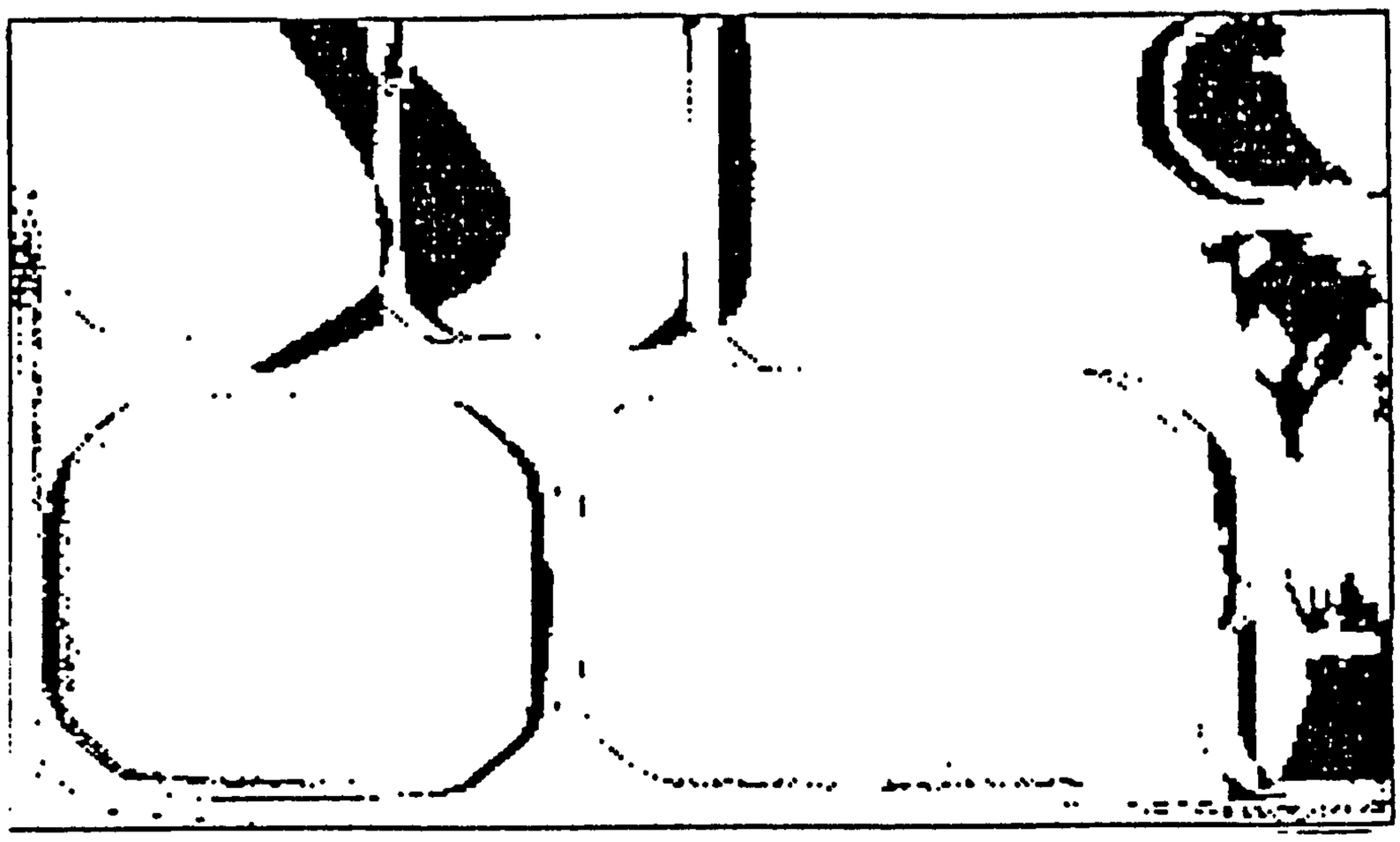


Figure 5.24a

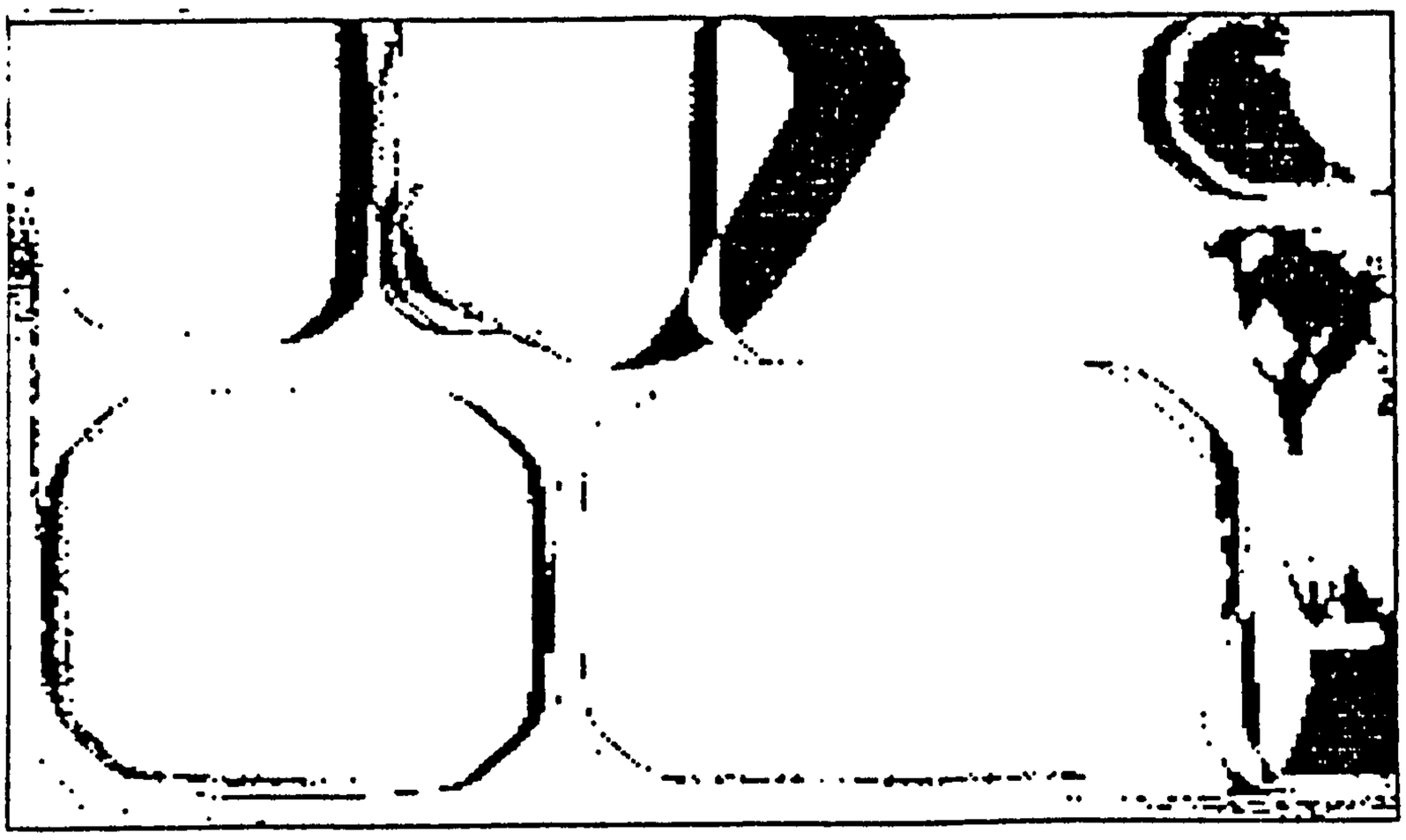


Figure 5.24b

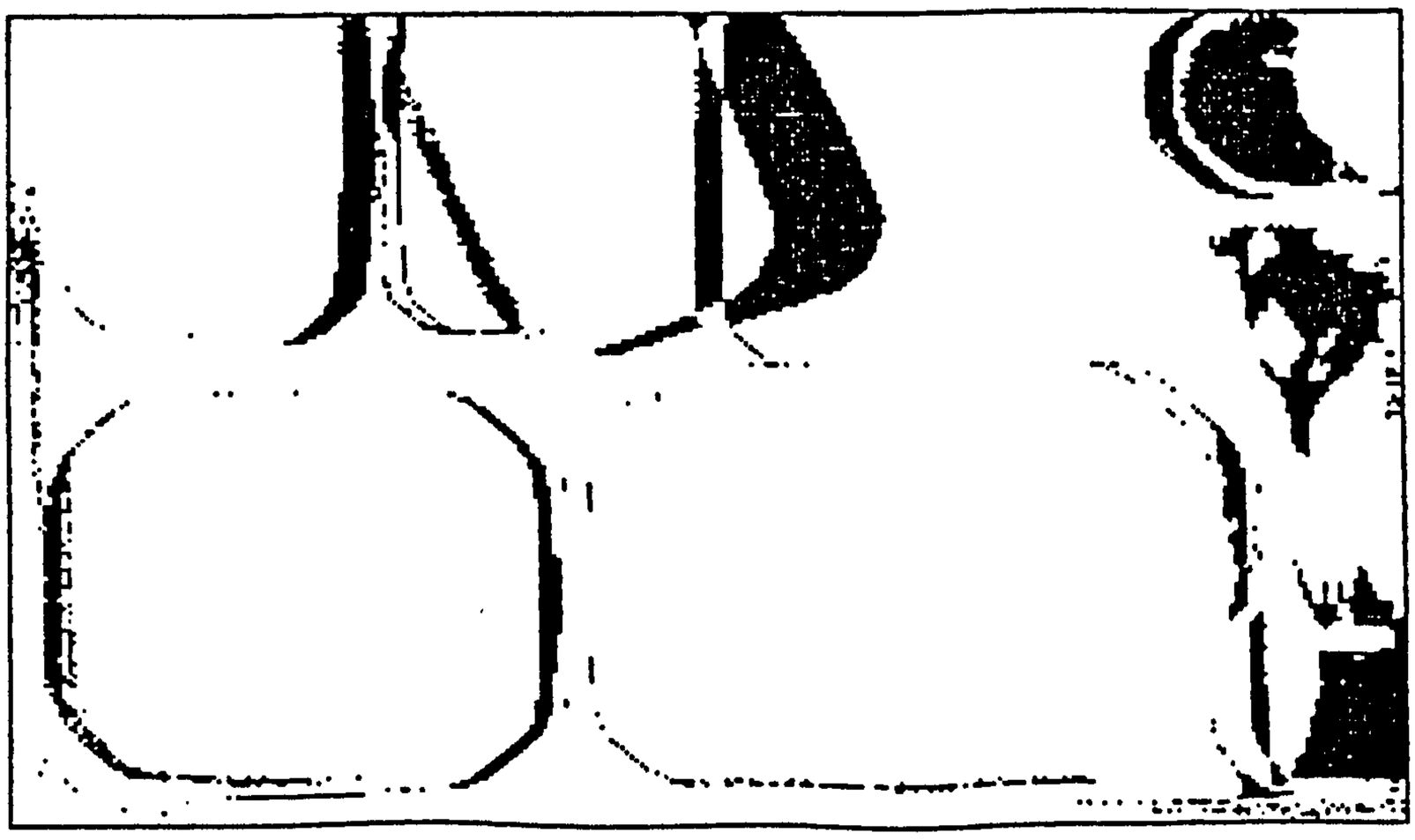


Figure 5.25a

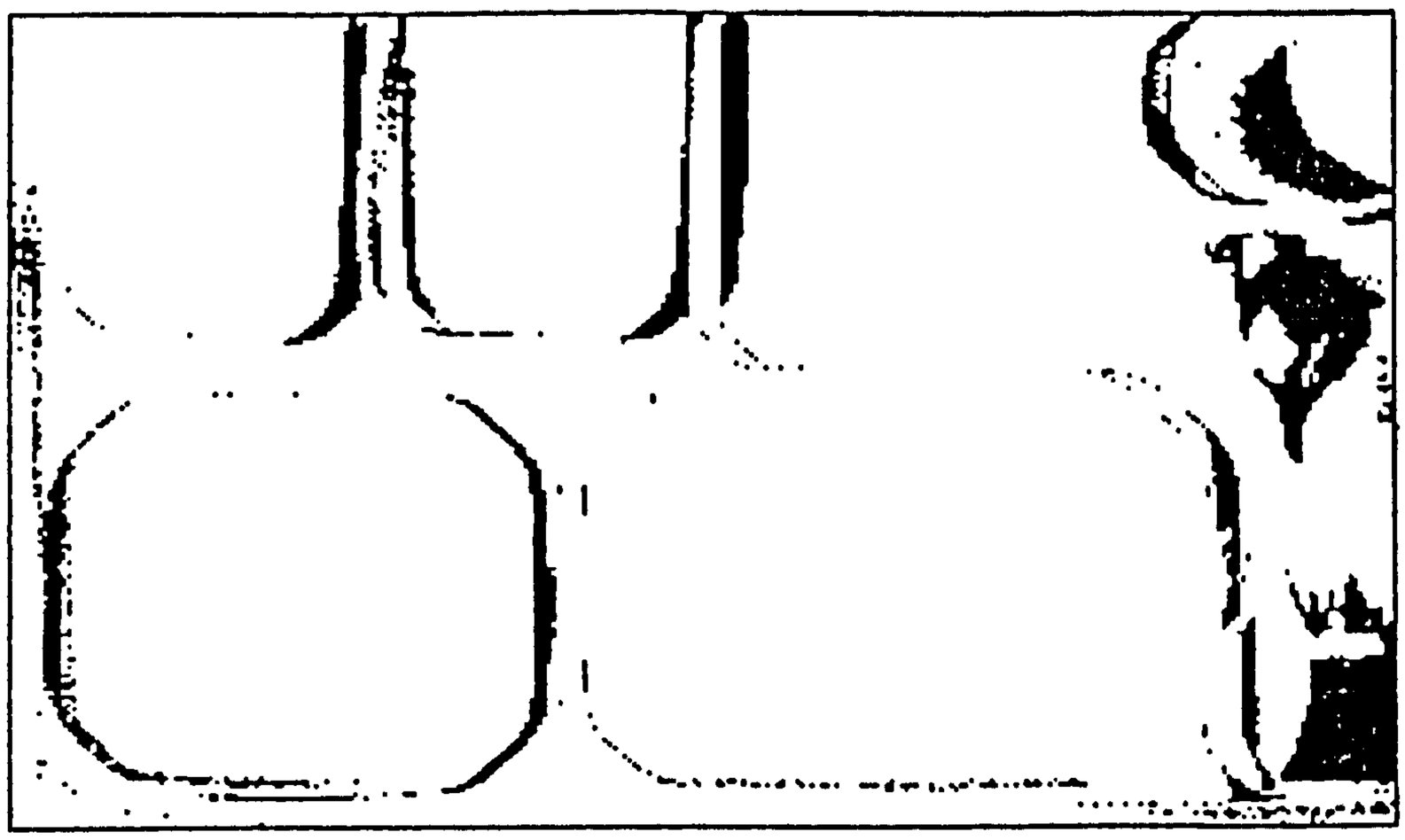


Figure 5.25b

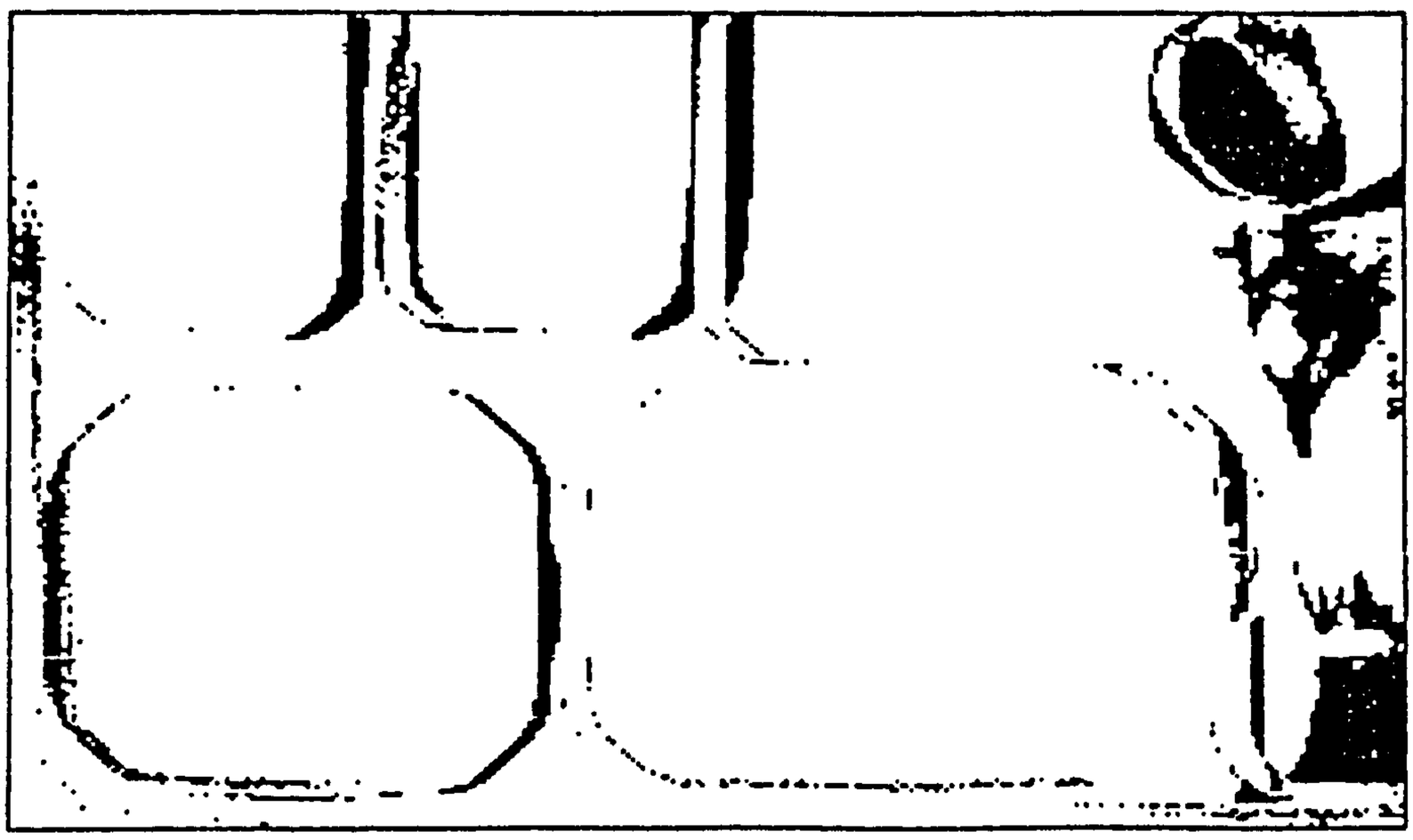


Figure 5.26a

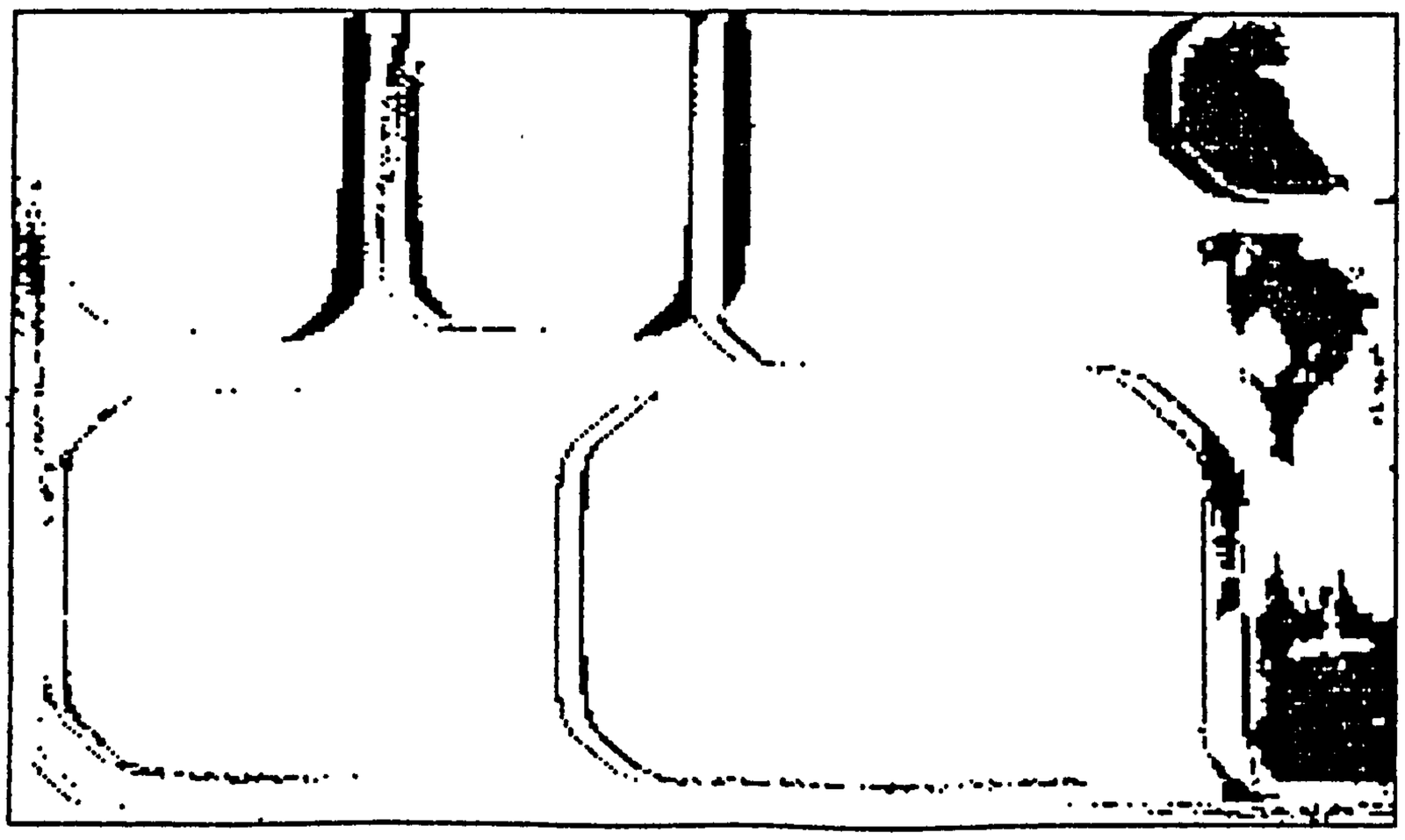


Figure 5.26b

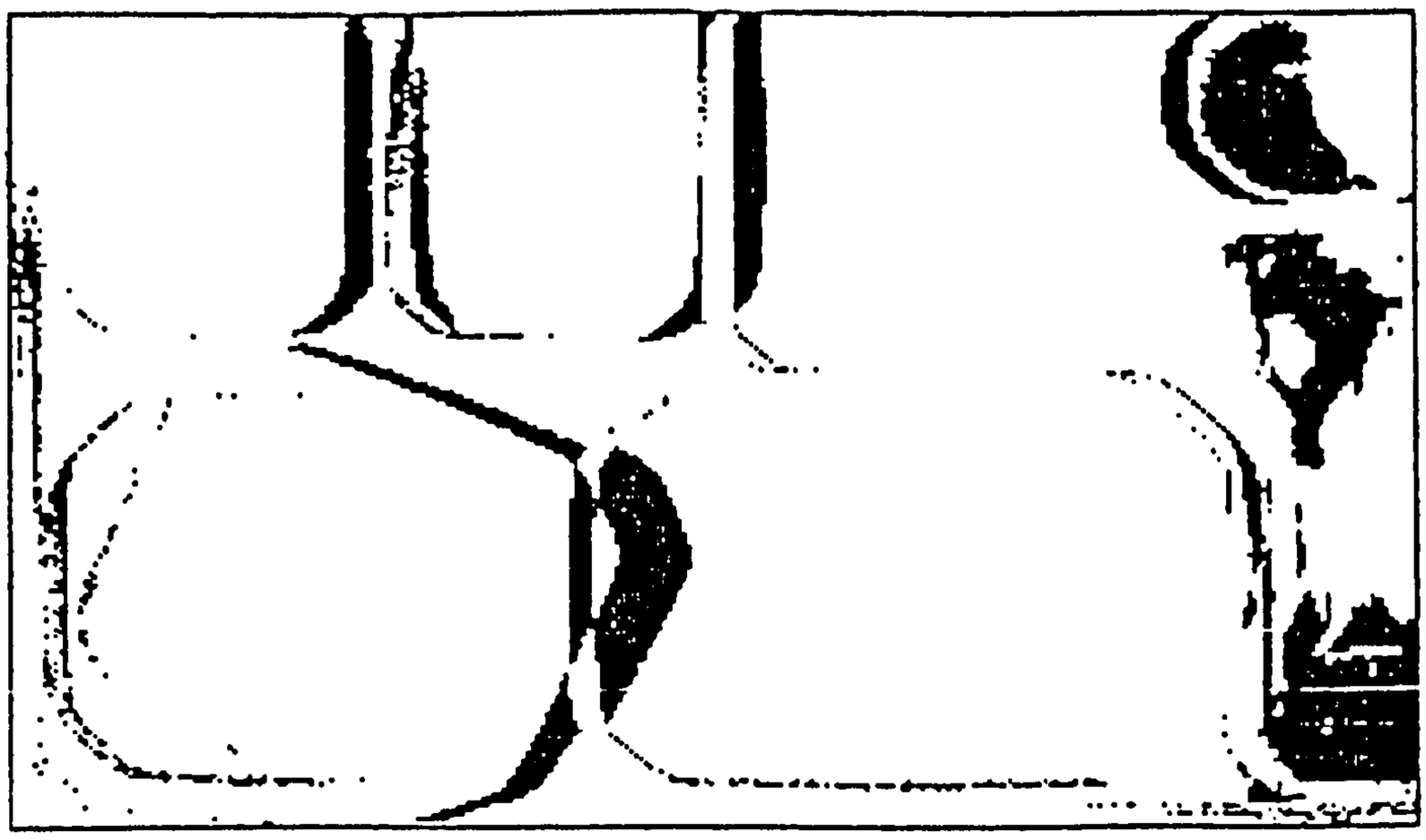


Figure 5.27a

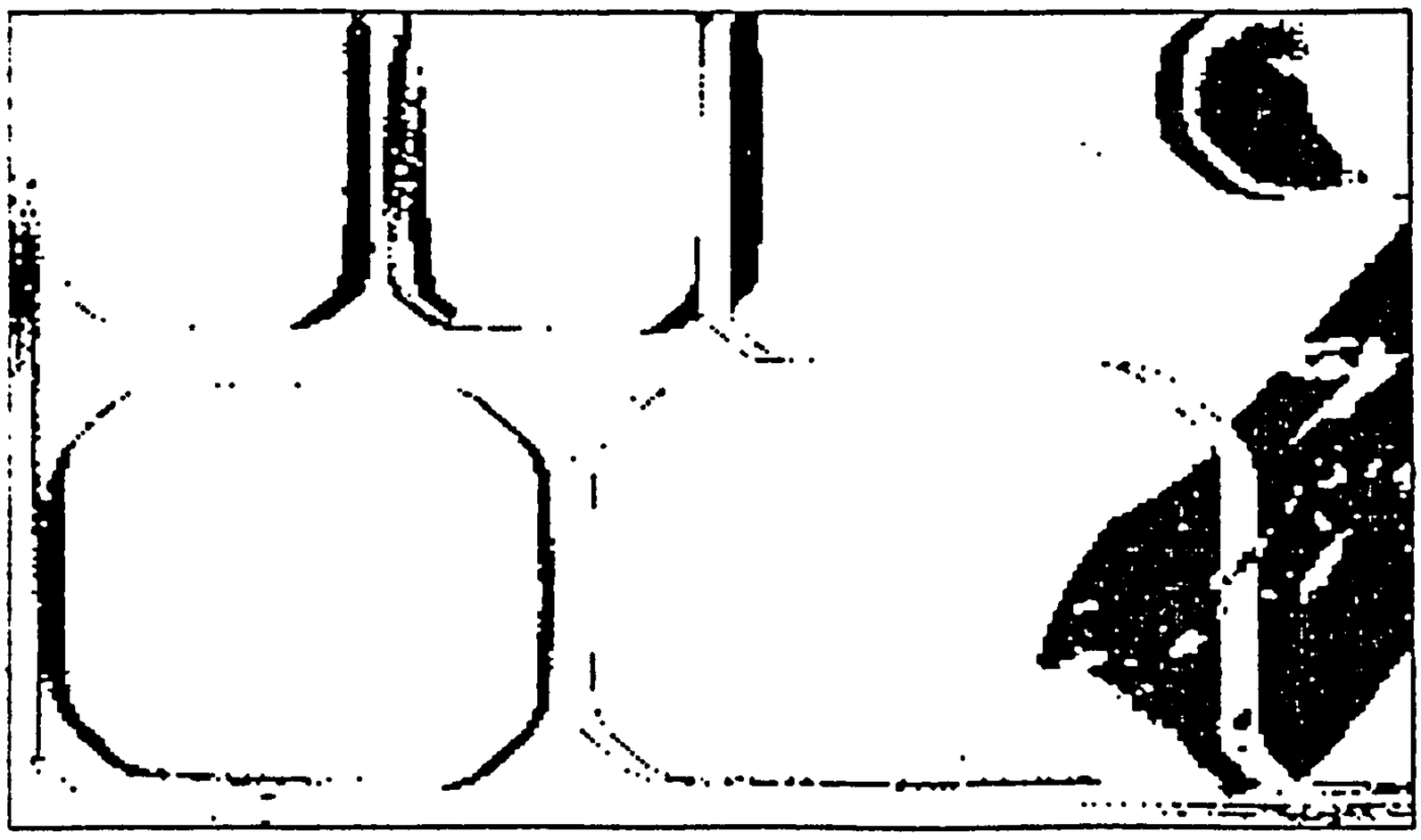


Figure 5.27b

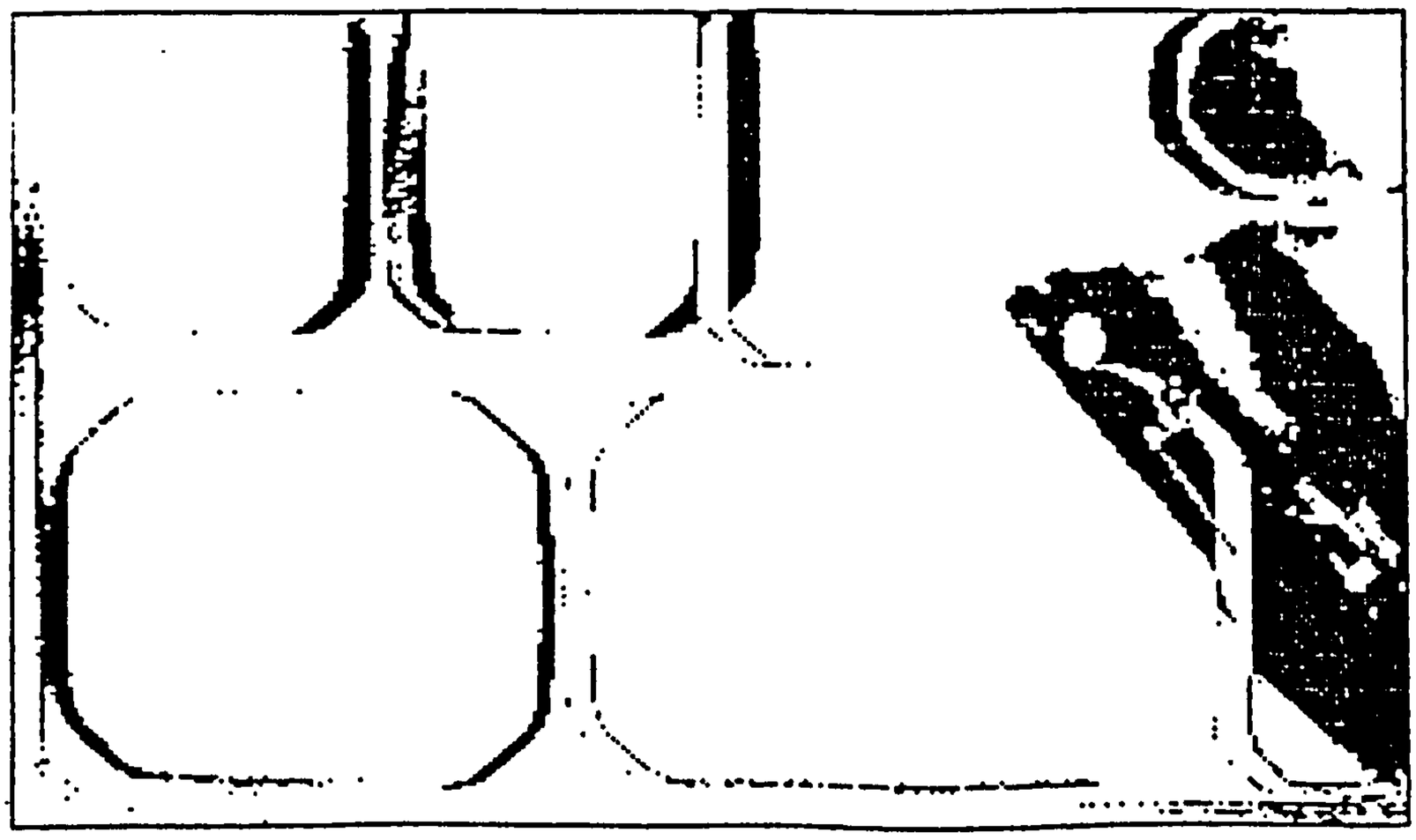


Figure 5.28

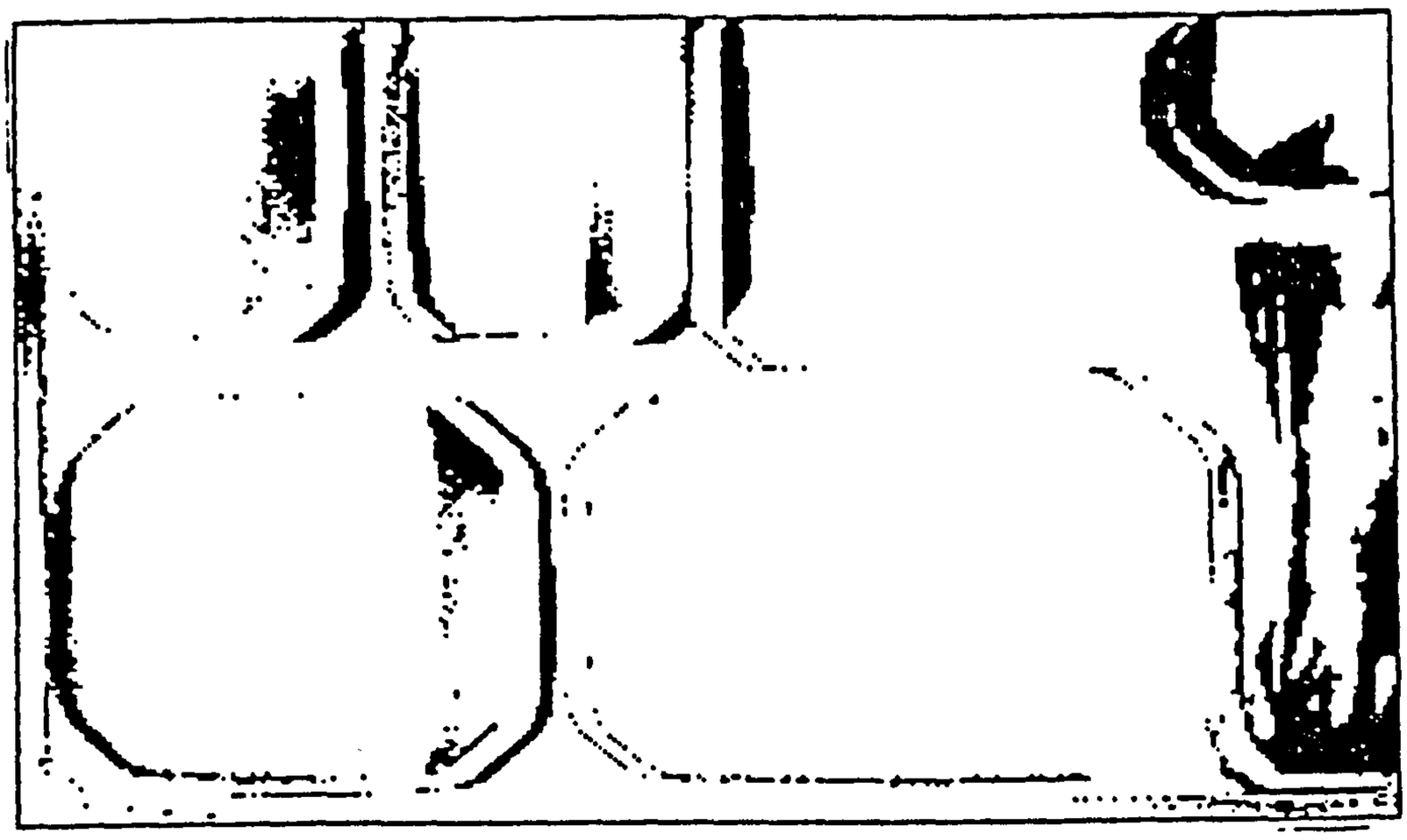


Figure 5.29

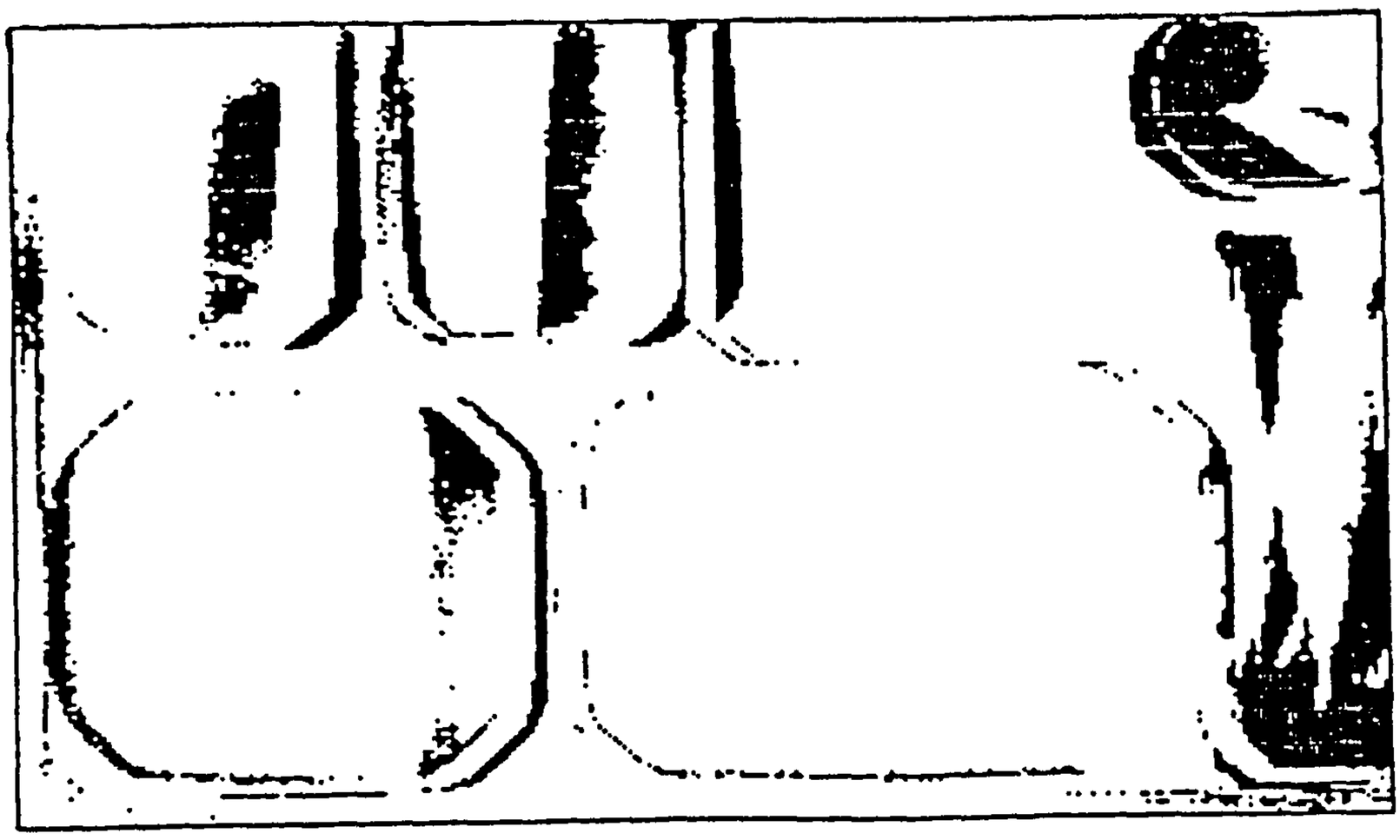
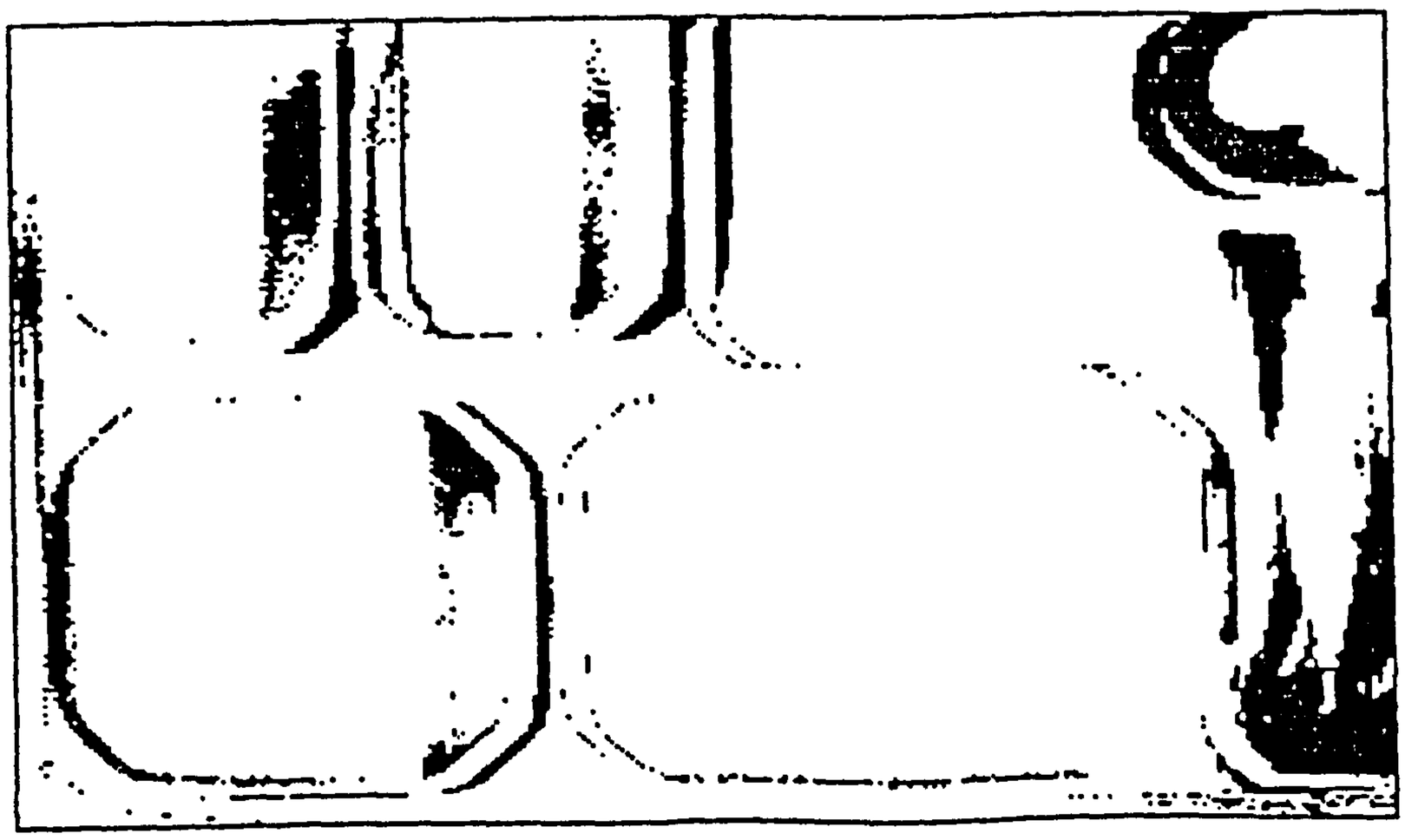


Figure 5.30



Schematic Representation of the Approach Adopted to the Second Vision Project.

Figure 6.1

Problem Specification.

The Problem.

To identify and pick-up a previously unknown, opaque item in random orientation on a flat surface.

Learning Sequence.

Generate a series of image files of an item, in all its anticipated stable 3D orientations.

Have image files of all objects to be processed been created?

No

Yes

Generate a feature-data file for each item.

Generate a cluster-data file for each item.

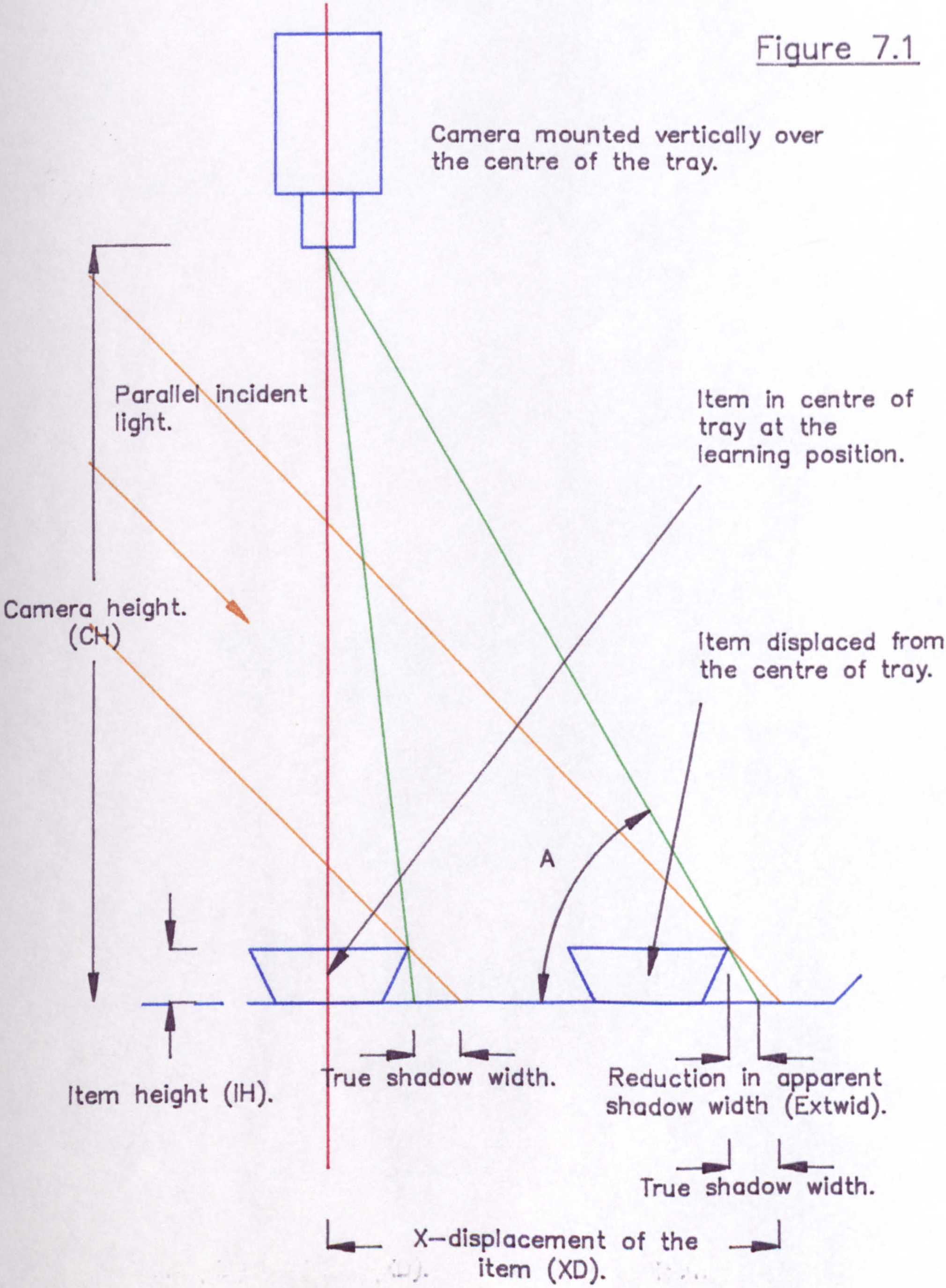
Image Testing Phase.

Image Testing Sequence.

1/ Set image processing options.
2/ Set process input and output options.
3/ Build an identification look-up table from the cluster-data files corresponding to the items to be identified and orientated.
4/ Place the sample at the inspection station.
5/ Take an image and process to extract features.
6/ Use features to generate a look-up table address.
7/ Object identification effected by reference to look-up table entry.
8/ Identified item pick-up points calculated.
9/ Pick-up coordinates supplied to external device.

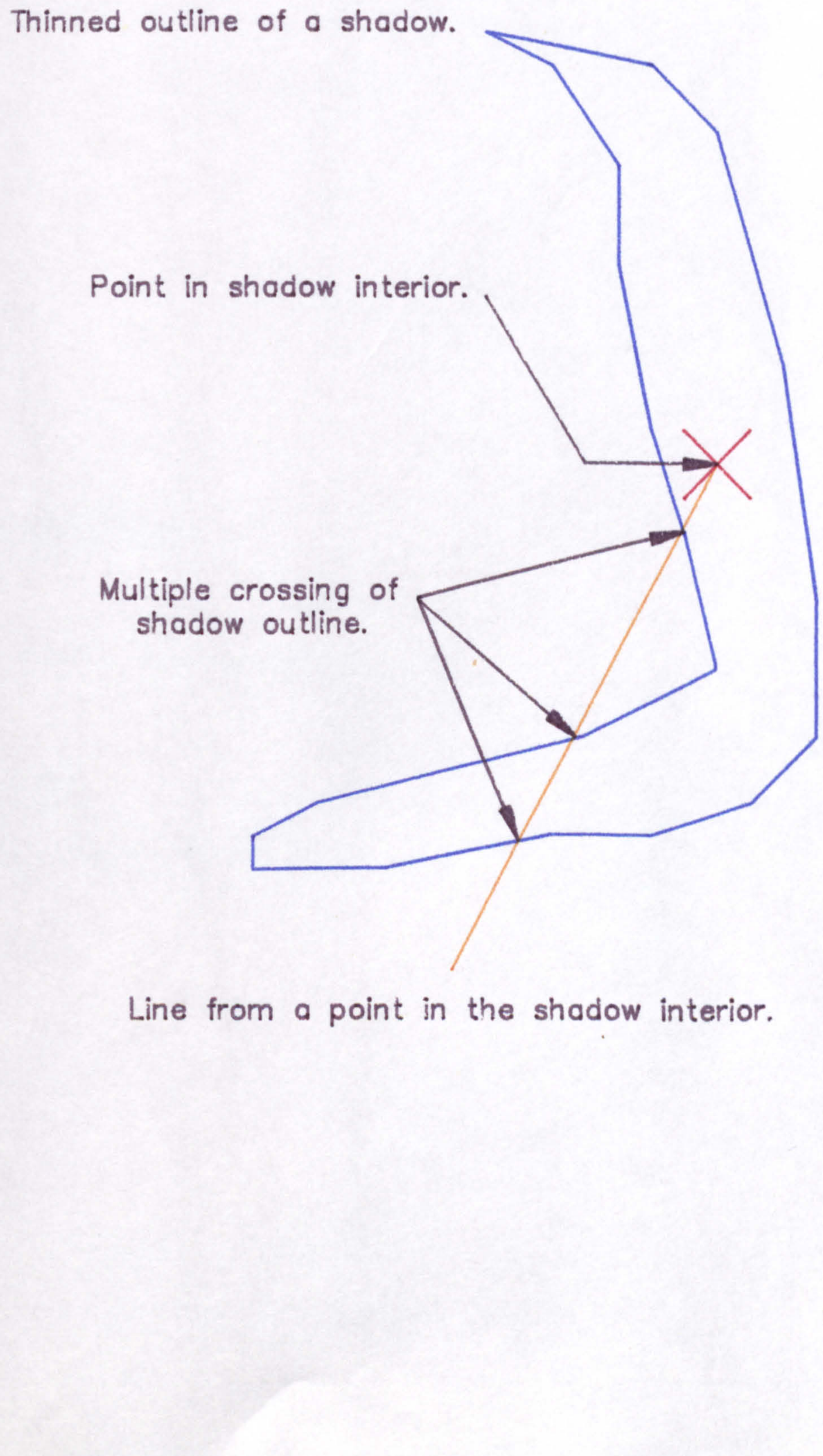
Diagram Illustrating the Change in Apparent Shadow Width as the Item Moves Away From the Centre of the Tray.

Figure 7.1



Schematic Diagram Illustrating a Multiple Crossing of a Shadow Outline.

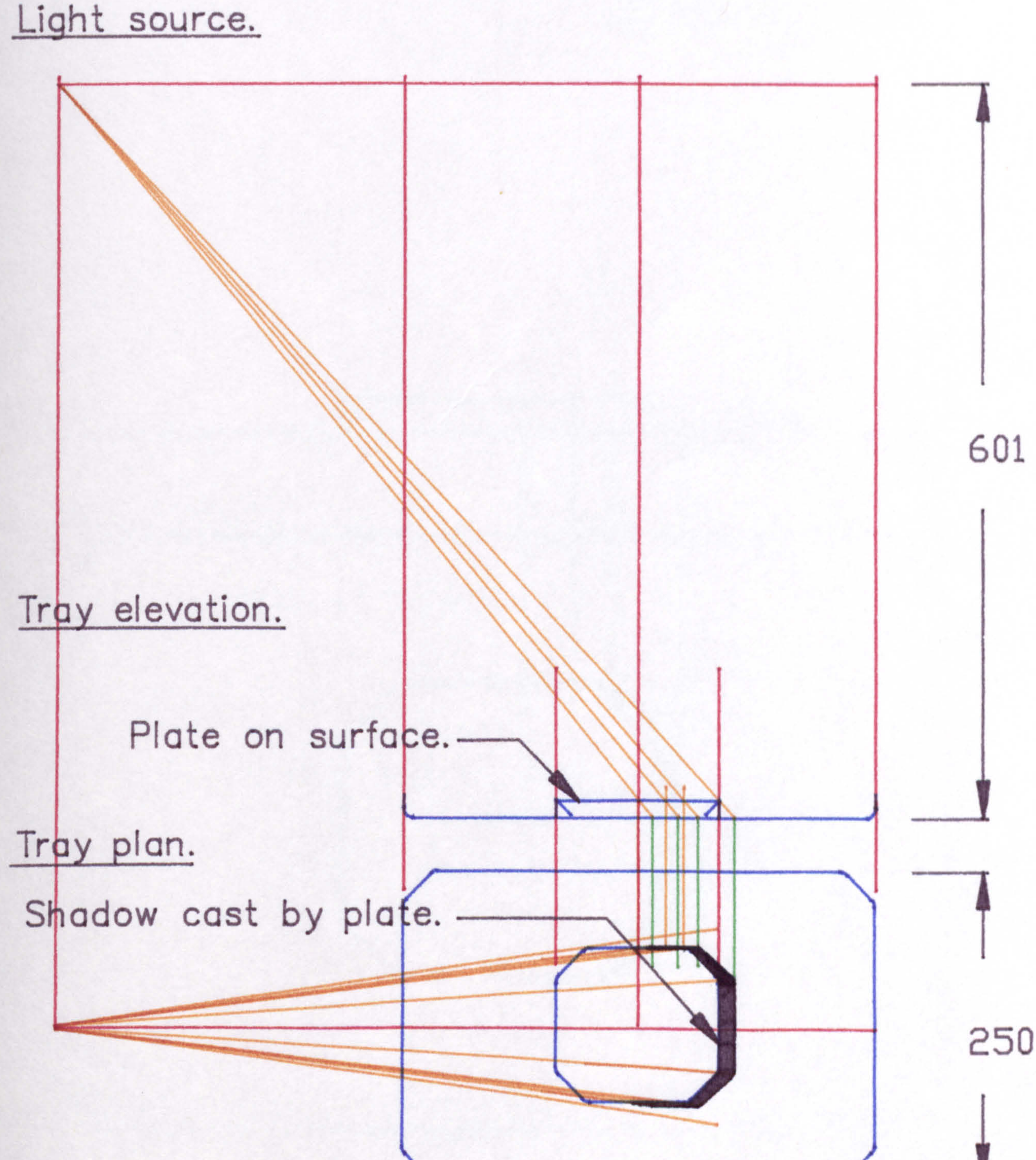
Figure 7.2



Line from a point in the shadow interior.

Schematic Diagram Illustrating the Shadow Cast by a Plate with an Aligned Axis.

Figure 7.3



(All dimensions in millimeters.)

TEXT BOUND INTO

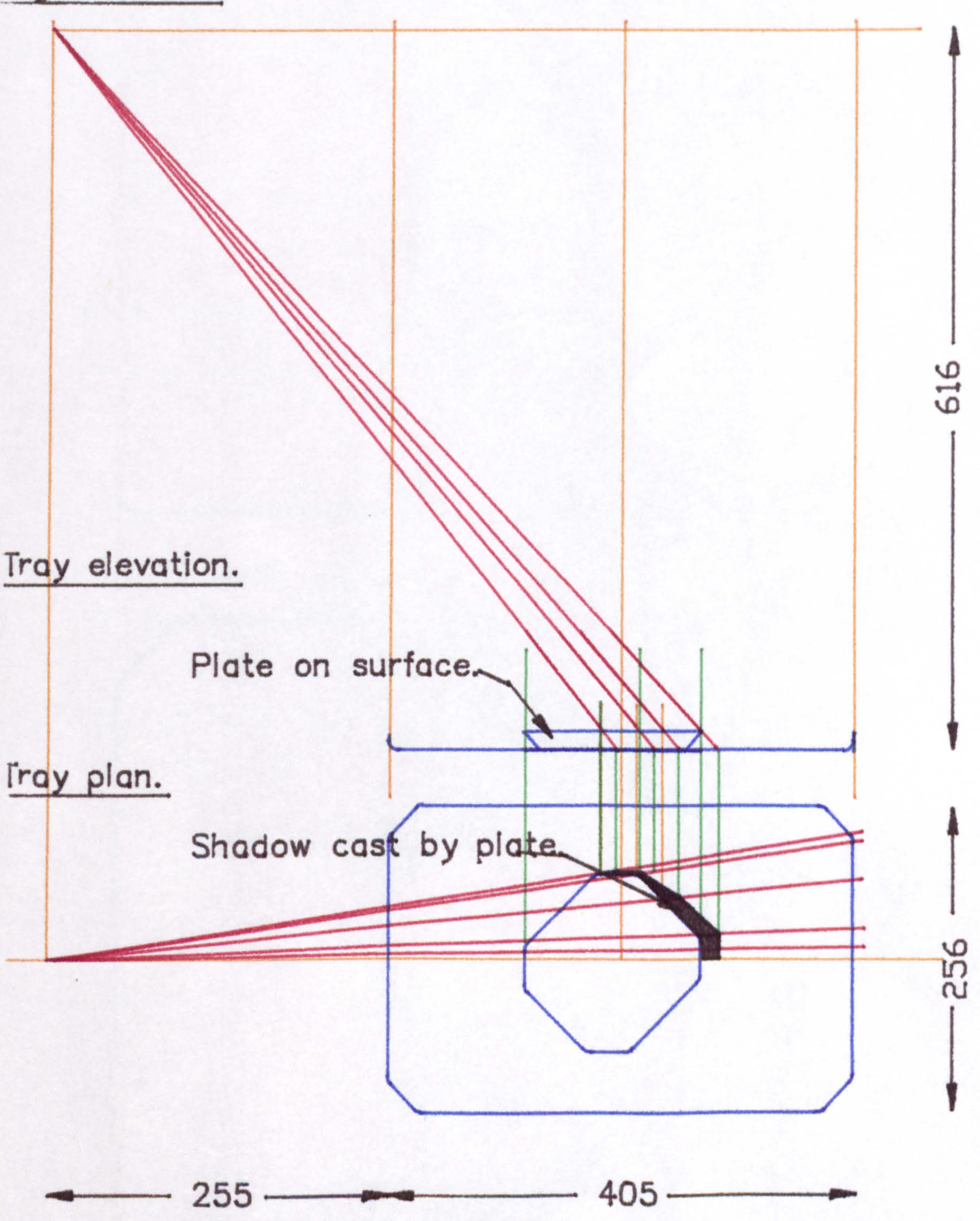
THE SPINE

Expanded View of Figure 7.3

Figure 7.3a

Schematic Diagram Illustrating the Shadow Cast by a Plate when Aligned Diagonally.

Figure 7.4



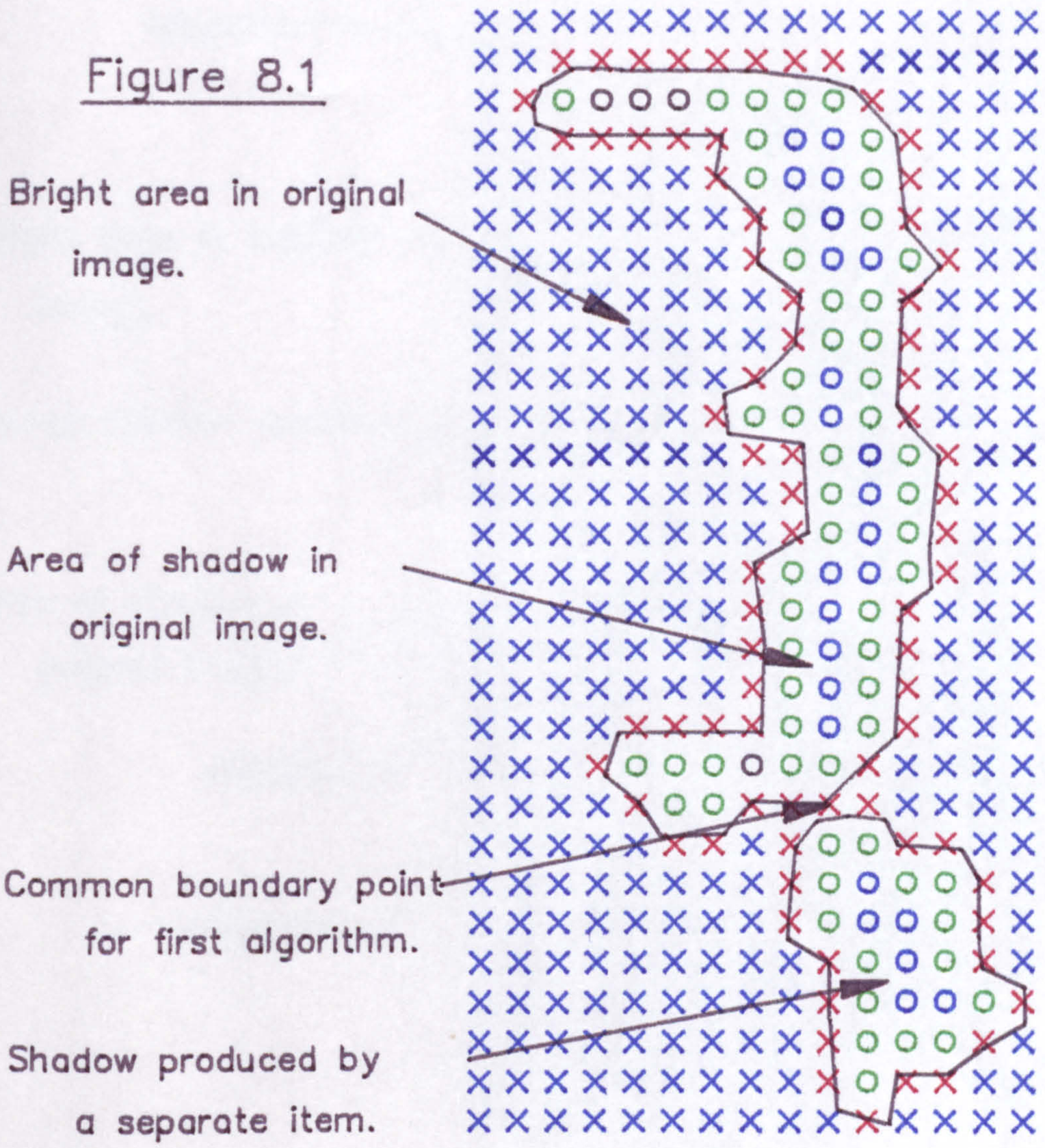
(All dimensions in millimeters.)

Expanded View of Figure 7.4

Figure 7.4a

Schematic Diagram Illustrating the Effect of Different Thinning Algorithms.

Figure 8.1



Key.

——— Original shadow outline.

× × Points originally digital one.

o o o Points originally digital zero.

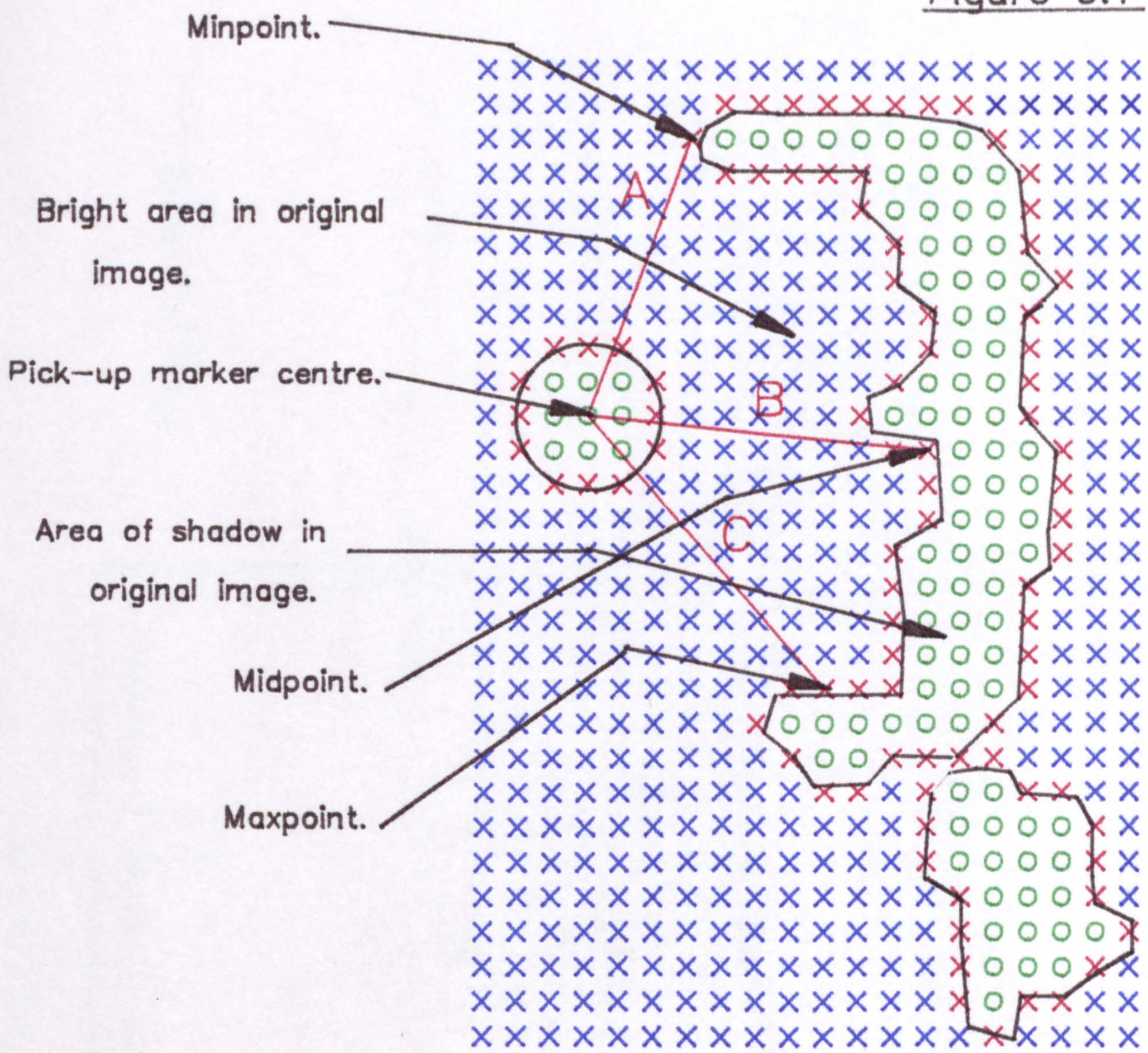
× Points remaining digital one after processing with the first algorithm.

o Points set to digital one by the second or third thinning algorithm.

o Points made digital one by the second thinning algorithm only.

Schematic Diagram Illustrating the Pick-up Point Features.

Figure 9.1



Key.

A — Distance to Minpoint feature.

B — Distance to Midpoint feature.

C — Distance to Maxpoint feature.

——— Original shadow outline.

X X — Points originally digital one.

O — Points originally digital zero.

X — Points remaining digital one after processing with the first thinning algorithm.

Plot of Theoretical Bowl Perimeter.

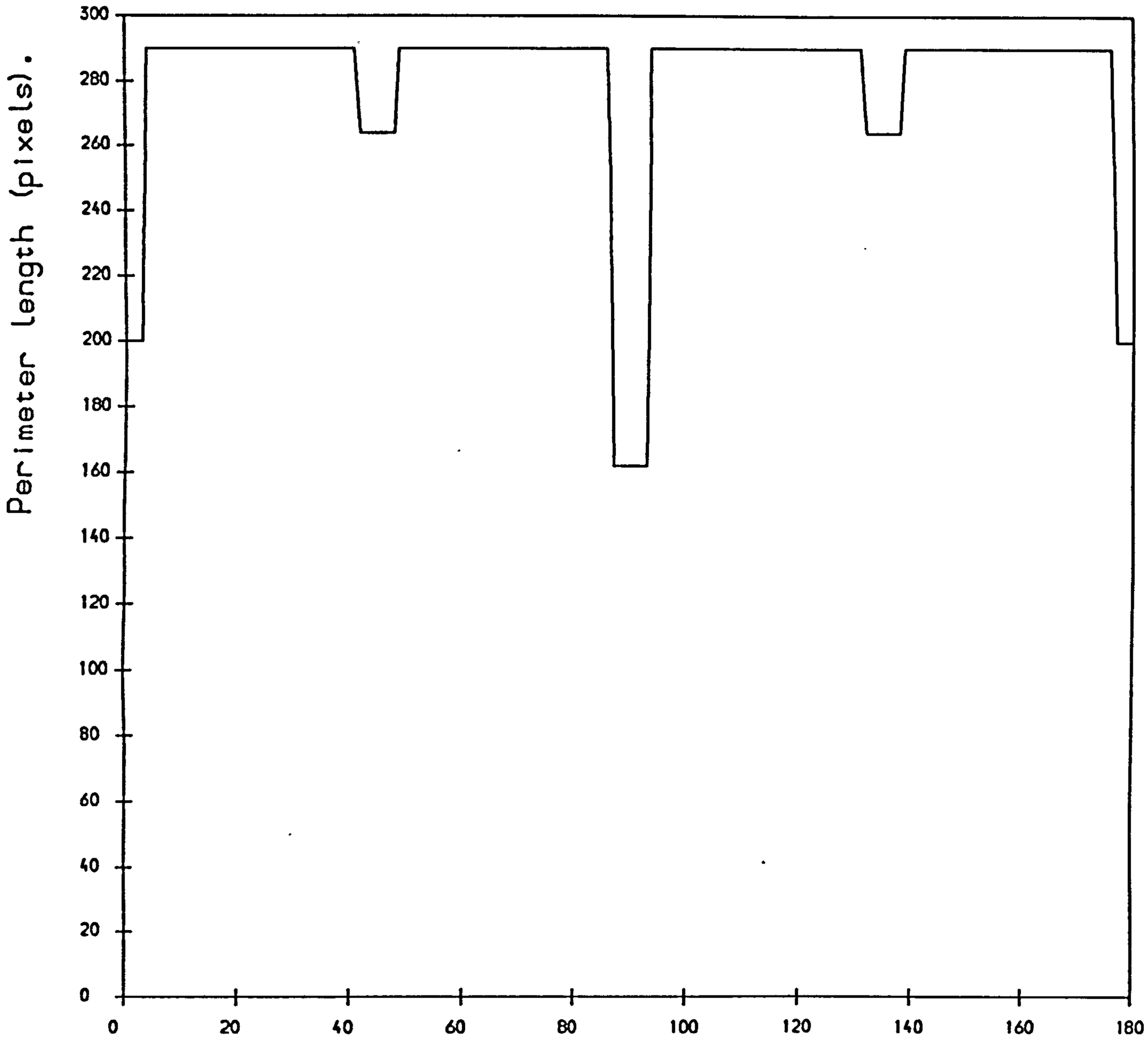


Figure 9.2

Plot of Theoretical Bowl Area.

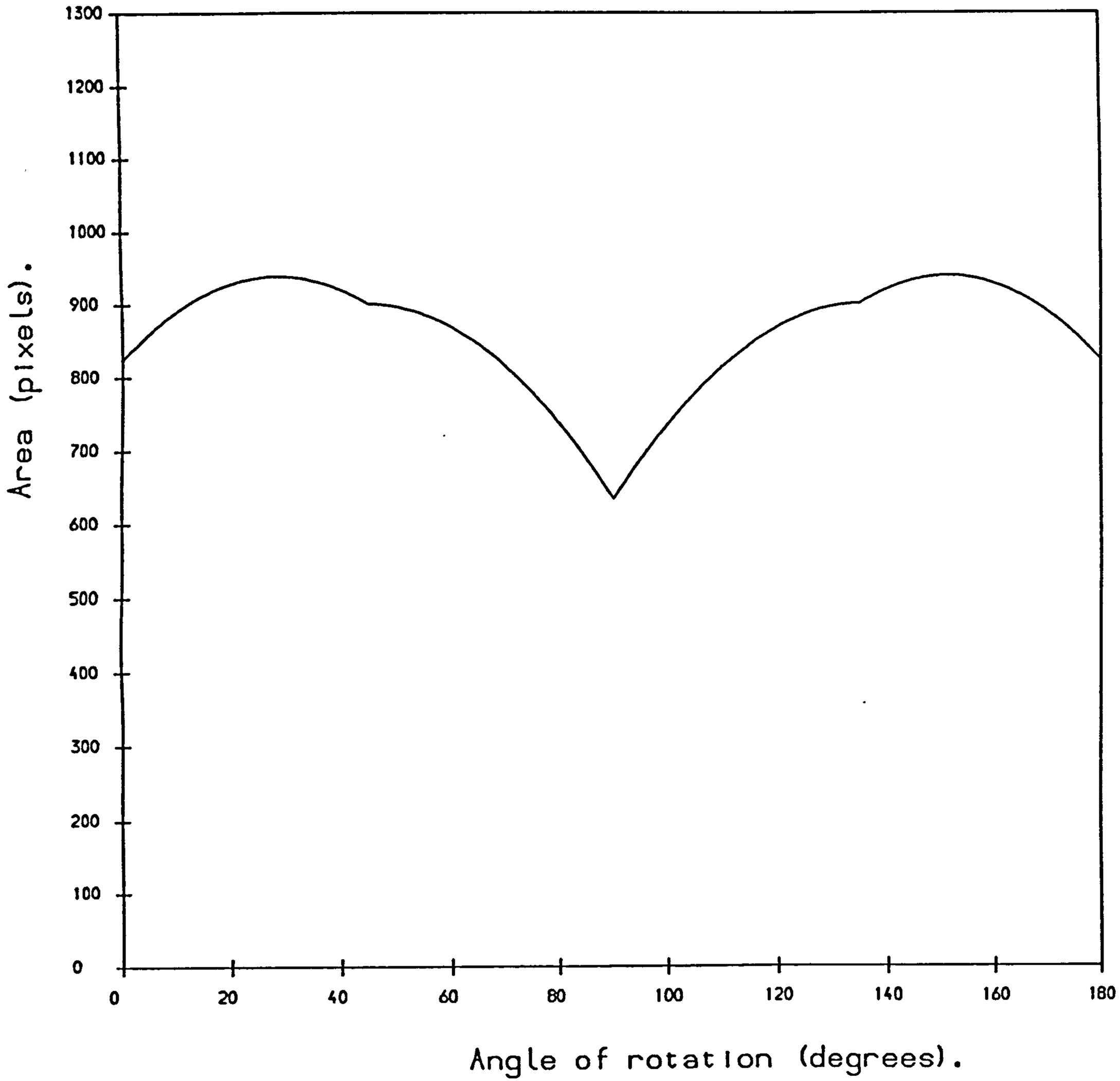


Figure 9.3

Bowl Perimeter as a Function of Angle of Rotation.

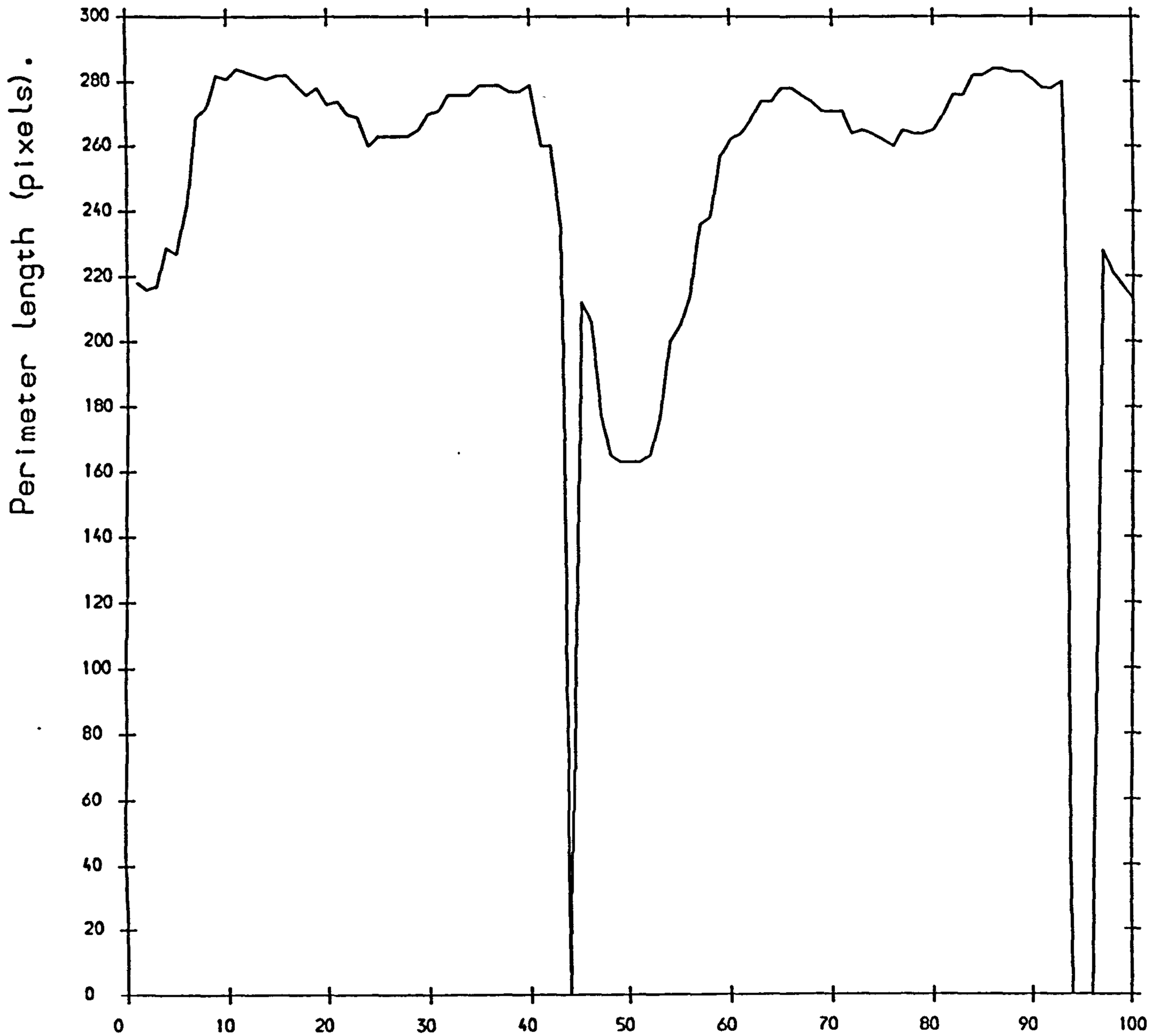


Figure 9.4

Bowl Area as a Function of Angle of Rotation.

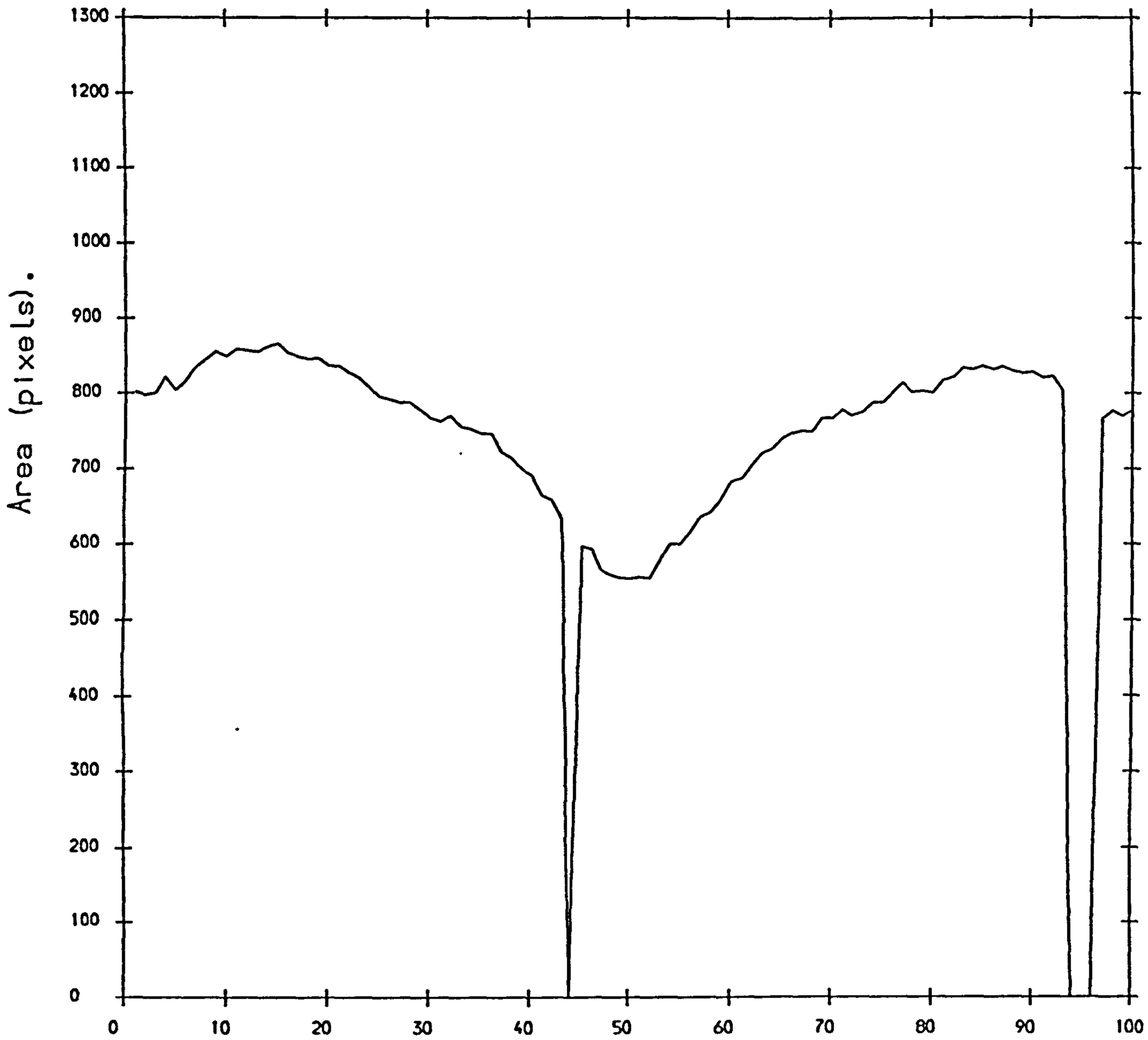


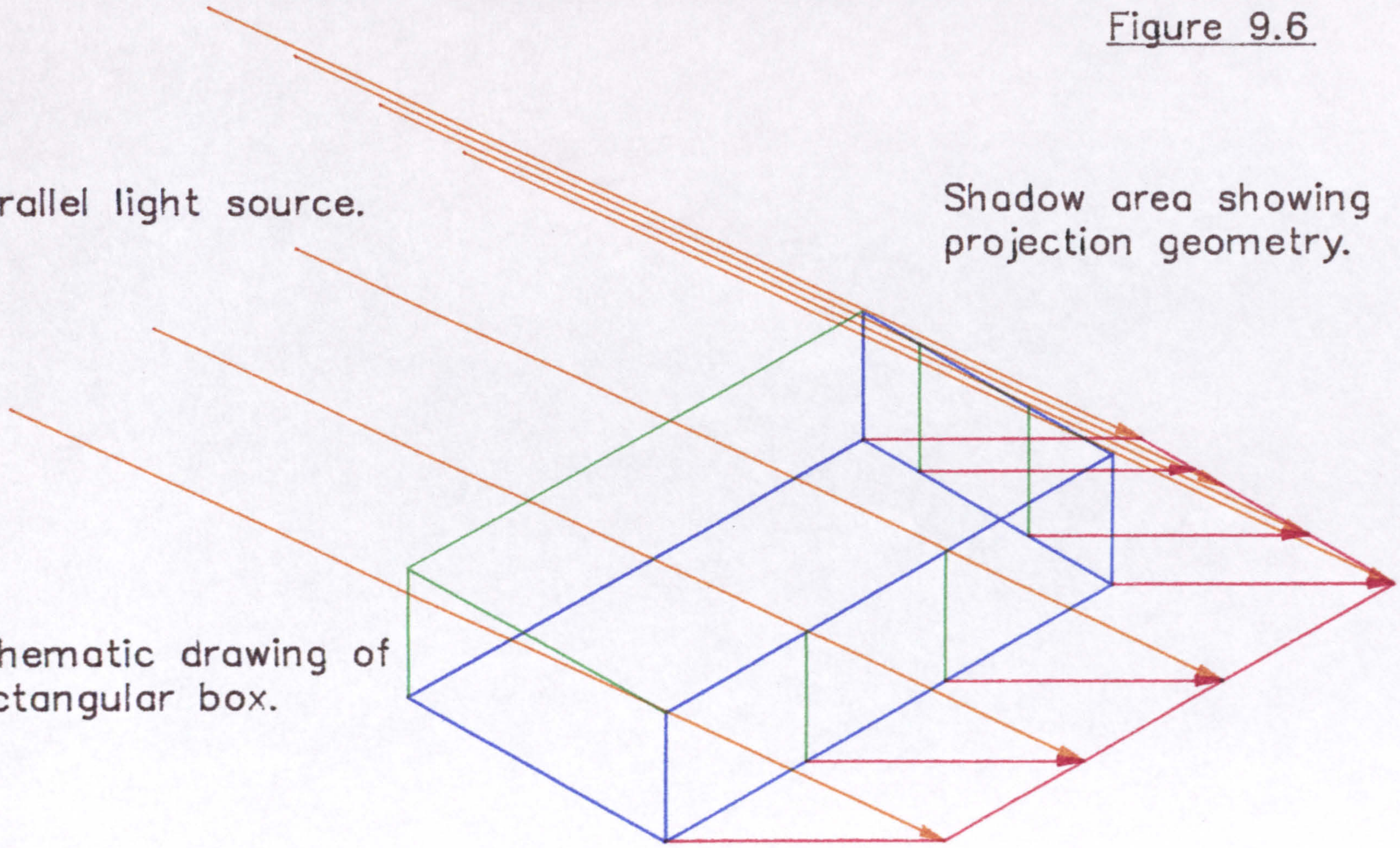
Figure 9.5

Schematic Diagram Illustrating the Principle of the Generation of the Theoretical Shadow Features.

Figure 9.6

Parallel light source.

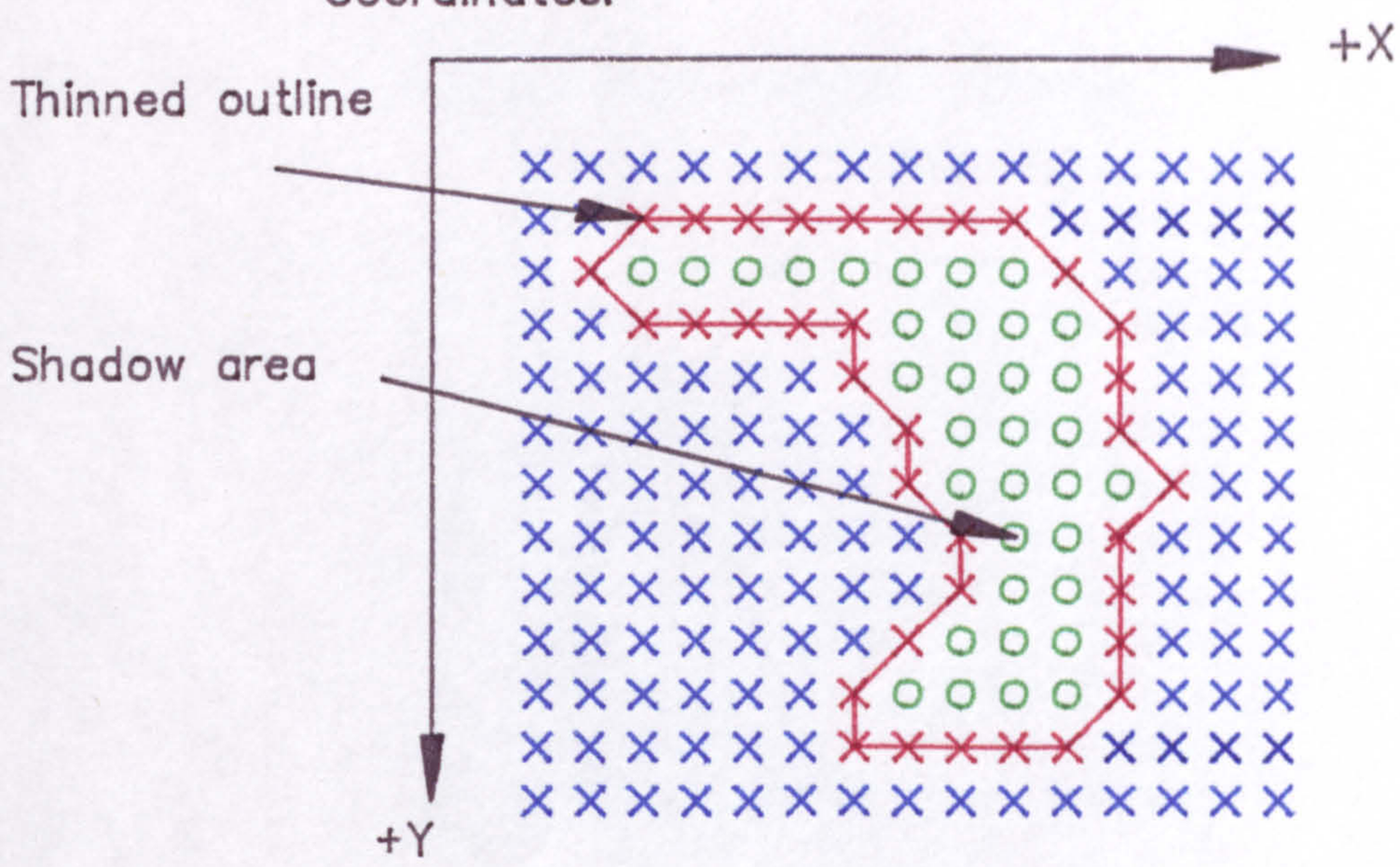
Shadow area showing projection geometry.



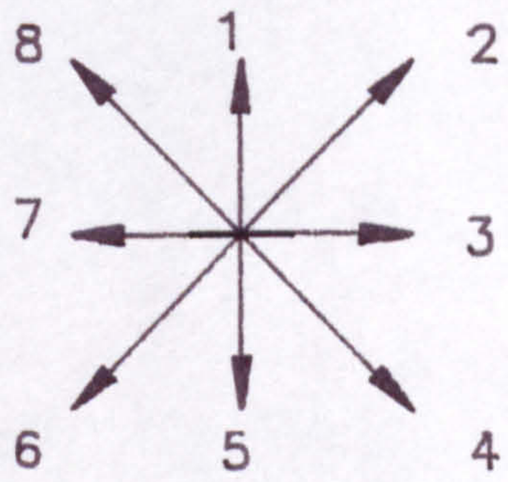
Schematic drawing of rectangular box.

Diagram Illustrating Shadow Area and Perimeter Calculation.

Figure 9.7



Direction vector coding.



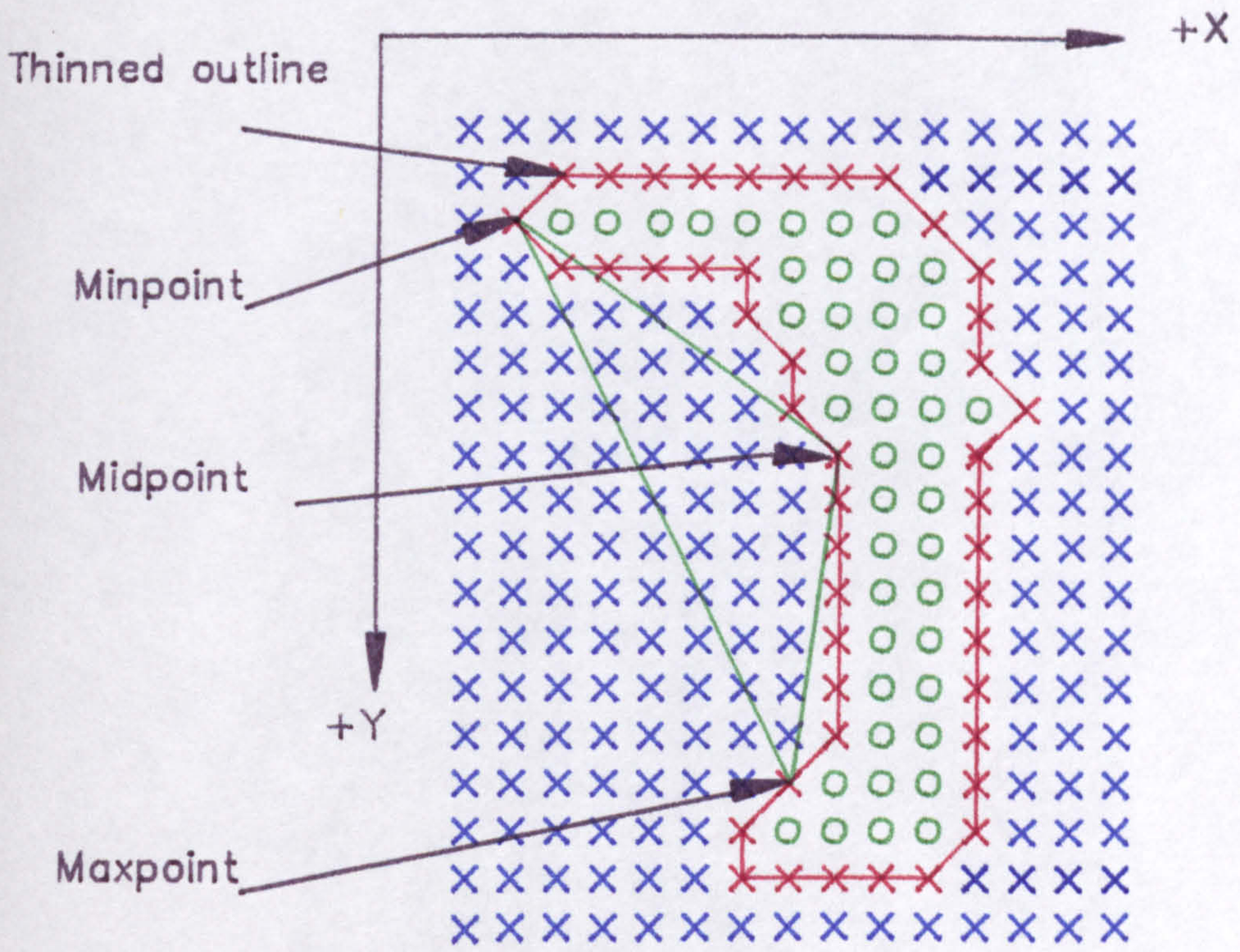
Direction vector contributions.

Direction	Perimeter contribution	Area contribution
1	1	0
2	$\sqrt{2}$	$(Y-\frac{1}{2}).1$
3	1	$Y.1$
4	$\sqrt{2}$	$(Y+\frac{1}{2}).1$
5	1	0
6	$\sqrt{2}$	$(Y+\frac{1}{2}).(-1)$
7	1	$Y.(-1)$
8	$\sqrt{2}$	$(Y-\frac{1}{2}).(-1)$

Diagram Illustating Alternative Features.

Figure 9.8

Schematic shadow image.



Min–max = Distance minpoint to maxpoint.

Min–mid = Distance minpoint to midpoint.

Mid–max = Distance midpoint to maxpoint.

Key

× × Points set at digital one in the original image.

o Points set at digital zero in the original image.

× Points remaining digital one in the image after the thinning operation.

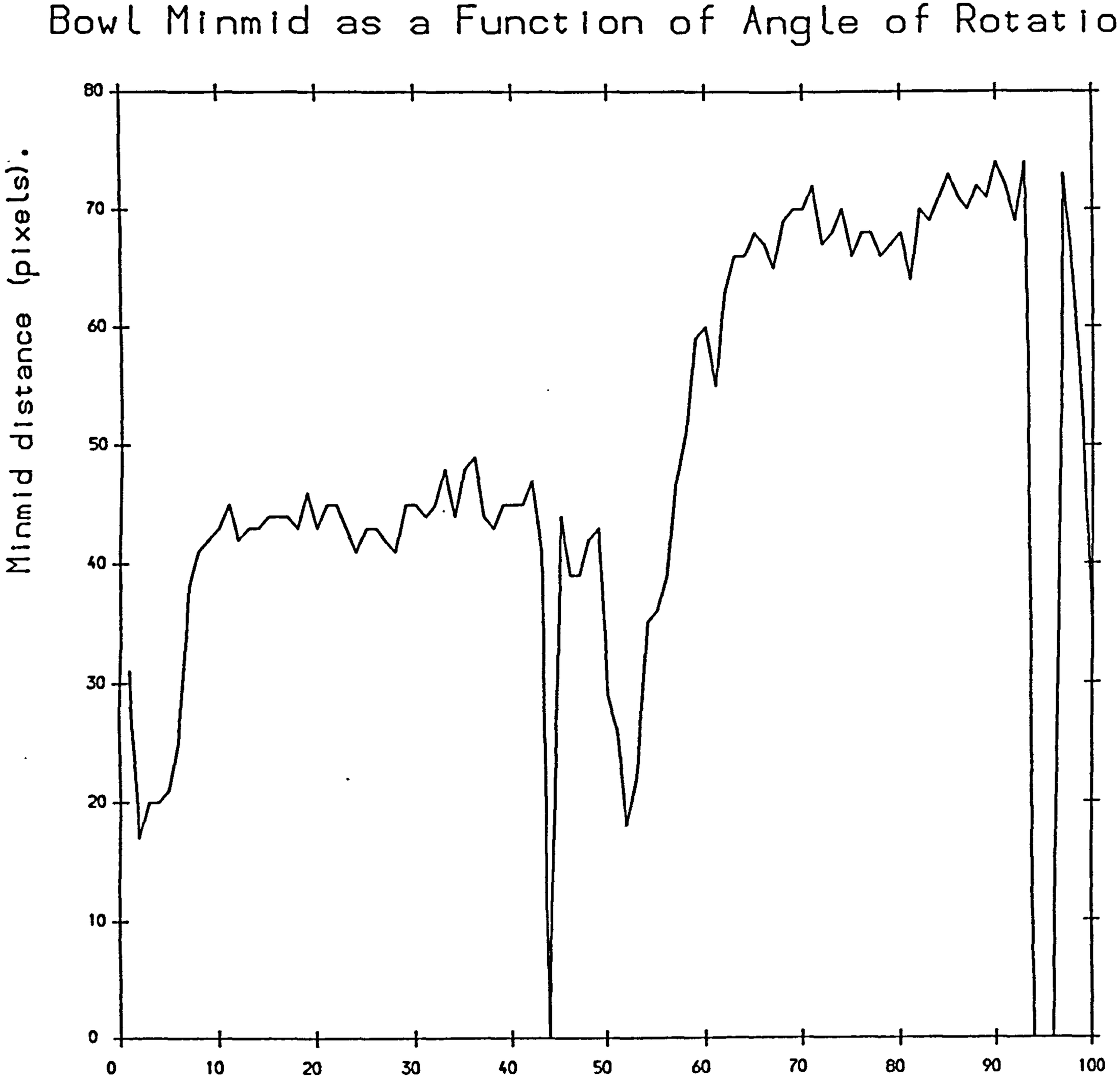

Bowl Minmid as a Function of Angle of Rotation.
Minmid distance (pixels).
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
Angle of rotation (units of 1.8 degrees).

Figure 9.9

Bowl Midmax as a Function of Angle of Rotation.

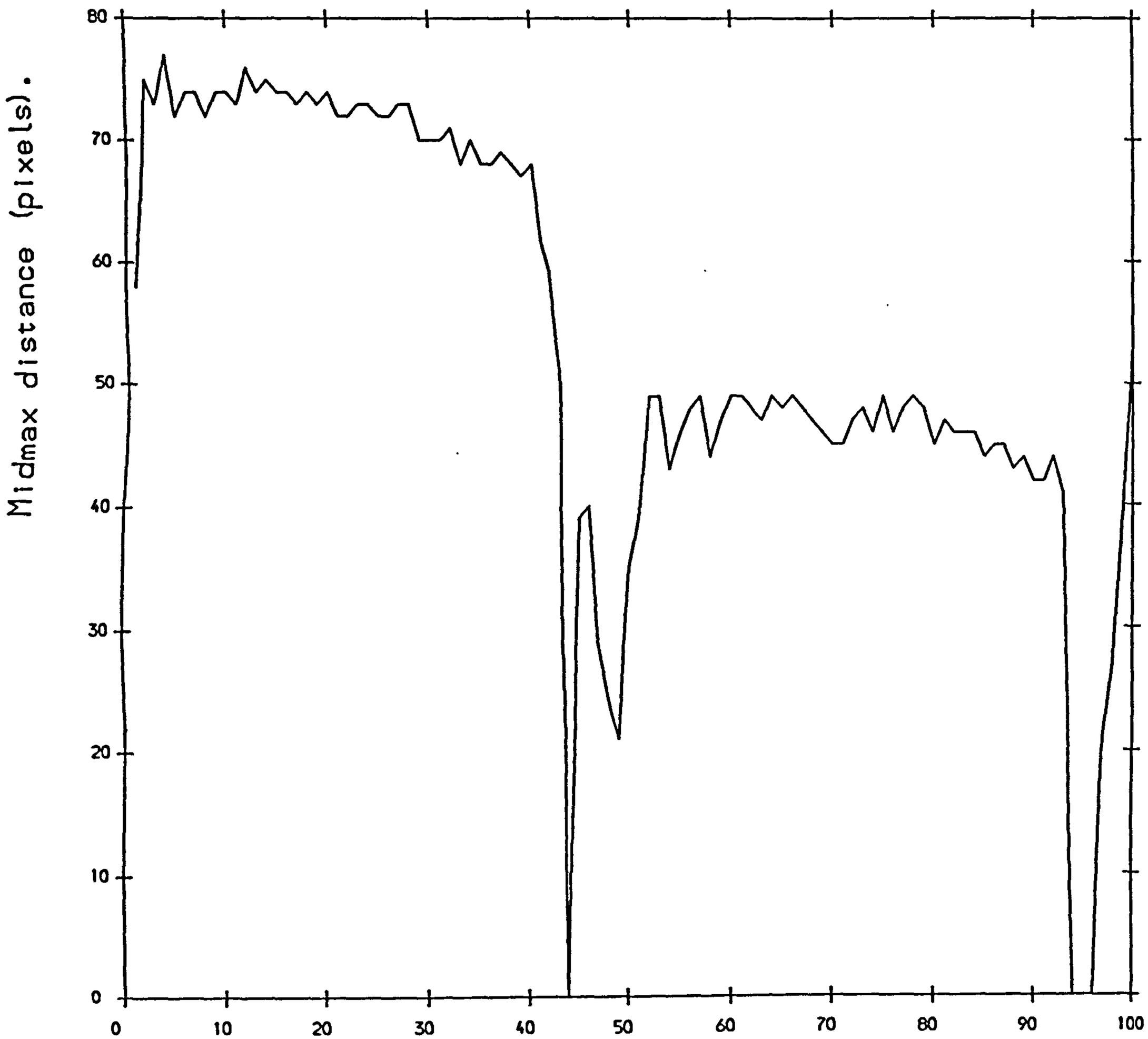
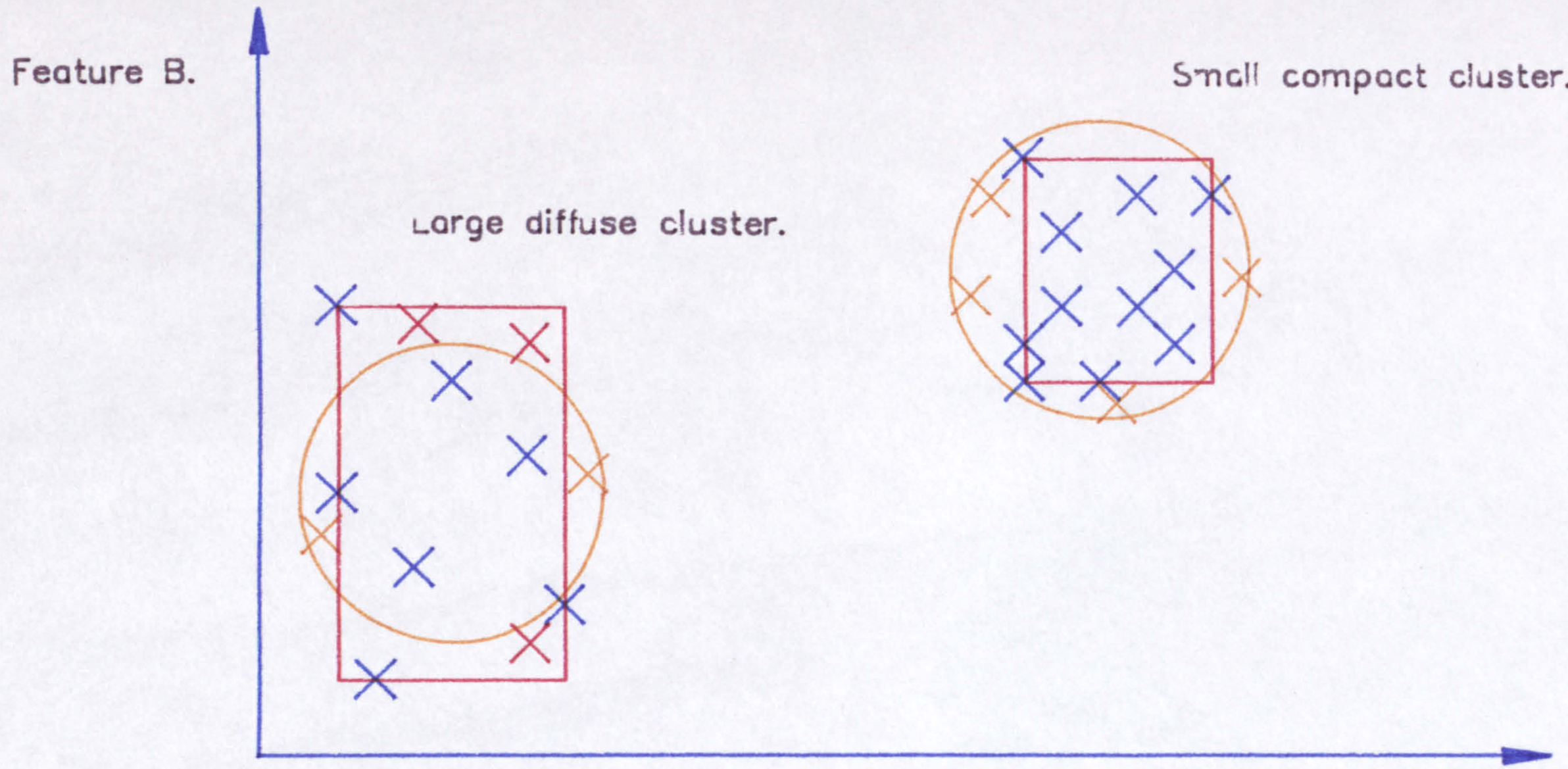


Figure 9.10

Diagram Illustrating the Problems Associated With Using a Fixed Distance–to–cluster–centre threshold criterion for object identification.

Figure 10.1



Key

- Feature vectors obtained in the object learning stage.
- Fixed distance from cluster centre diameters.
- The used rectangularized cluster sizes.
- Cases that would be incorrectly identified as being members of clusters using the fixed distance threshold.
- Cases that would not be identified as being members of clusters by using the fixed distance threshold.

APPLE PROGRAM FOR UNSUPERVISED VISION SYSTEM.

Figure 10.2

Set-up Phase.

Initialize Hardware.

1/ Enable Video Digitizer.
2/ Initialize parallel image transfer interface.
3/ Initialize stepper-motor interface.

Enter image capture parameters.

1/ Set rotatary table in correct starting position.
2/ Adjust binary image capture threshold.
3/ Enter the number of images to be captured.
4/ Enter the number of 0.9 deg. steps of table rotation between each image captured.

Sampling Phase.

Digitize an image into hi-res graphics page 1.

Wait state entered.

Transfer image via parallel interface to the DUMC 68020 unit.

Have all specified images been captured?

No

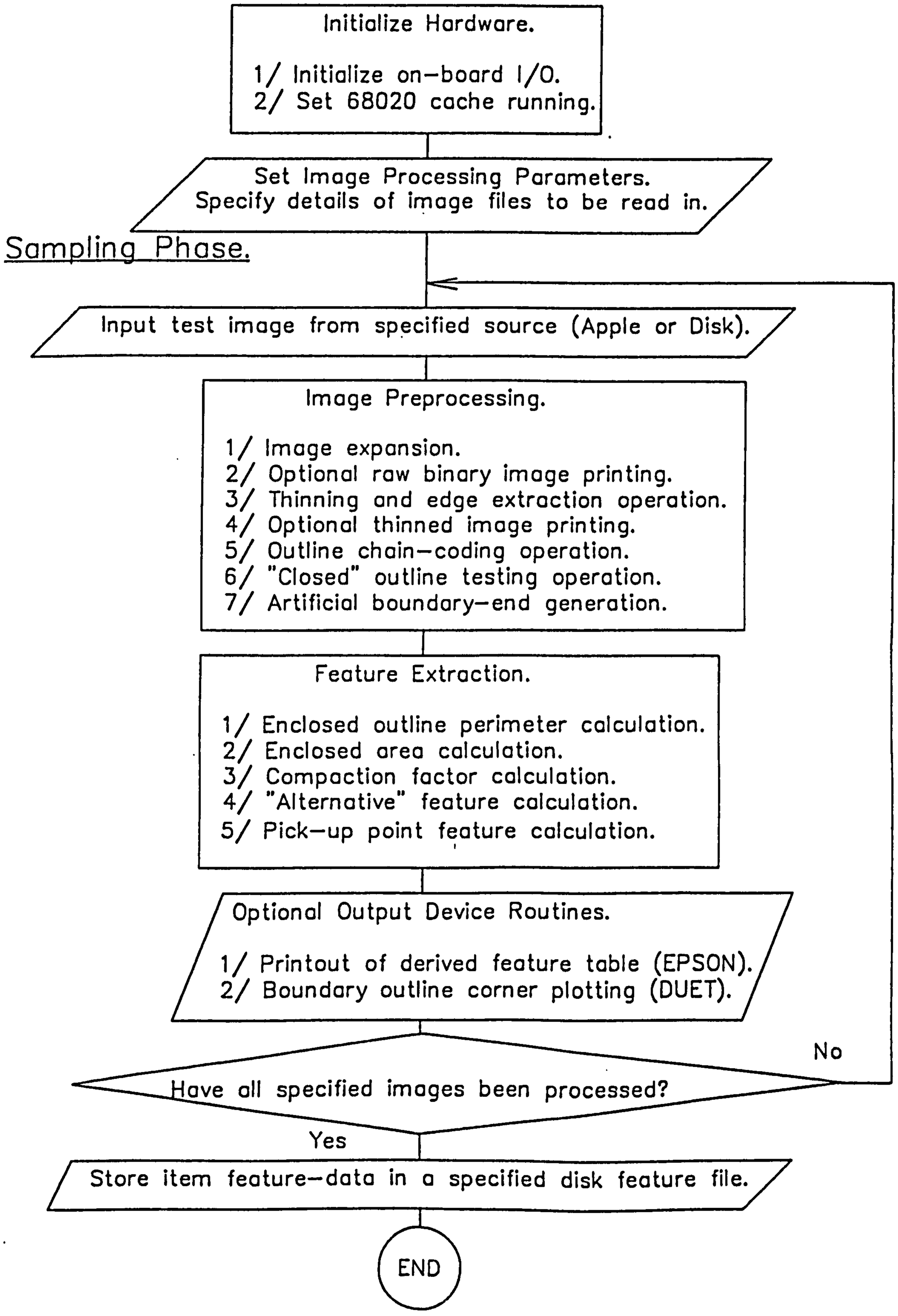
Rotate object on rotary table.

Yes

END

DUMC UNIT OBJECT LEARNING PROGRAM.

Figure 10.3



Bowl Compaction as a Function of Angle of Rotation.

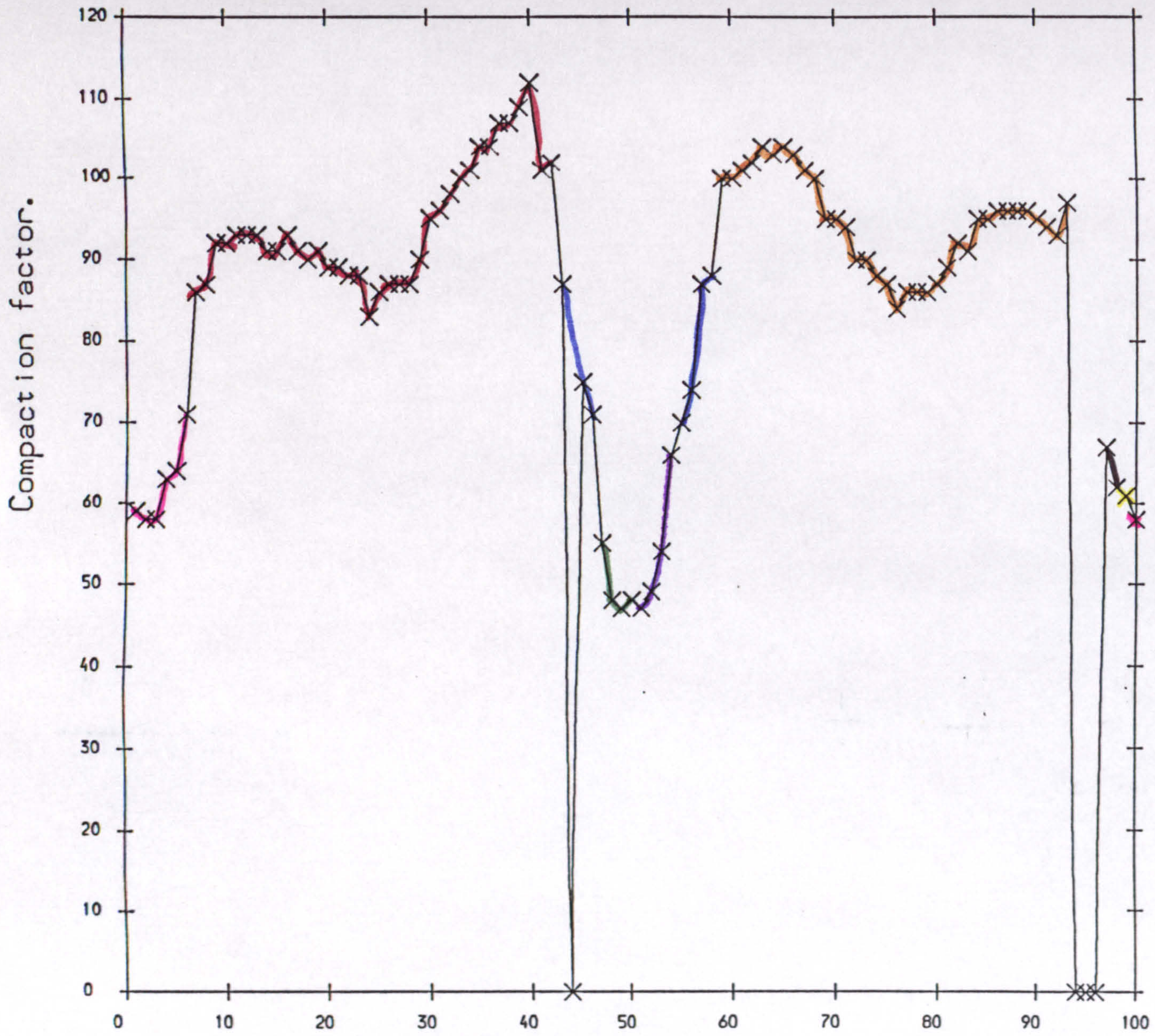


Figure 10.4

Key to Clusters

- Cluster 0
- Cluster 1
- Cluster 2
- Cluster 3
- Cluster 4
- Cluster 5
- Cluster 6
- Cluster 7

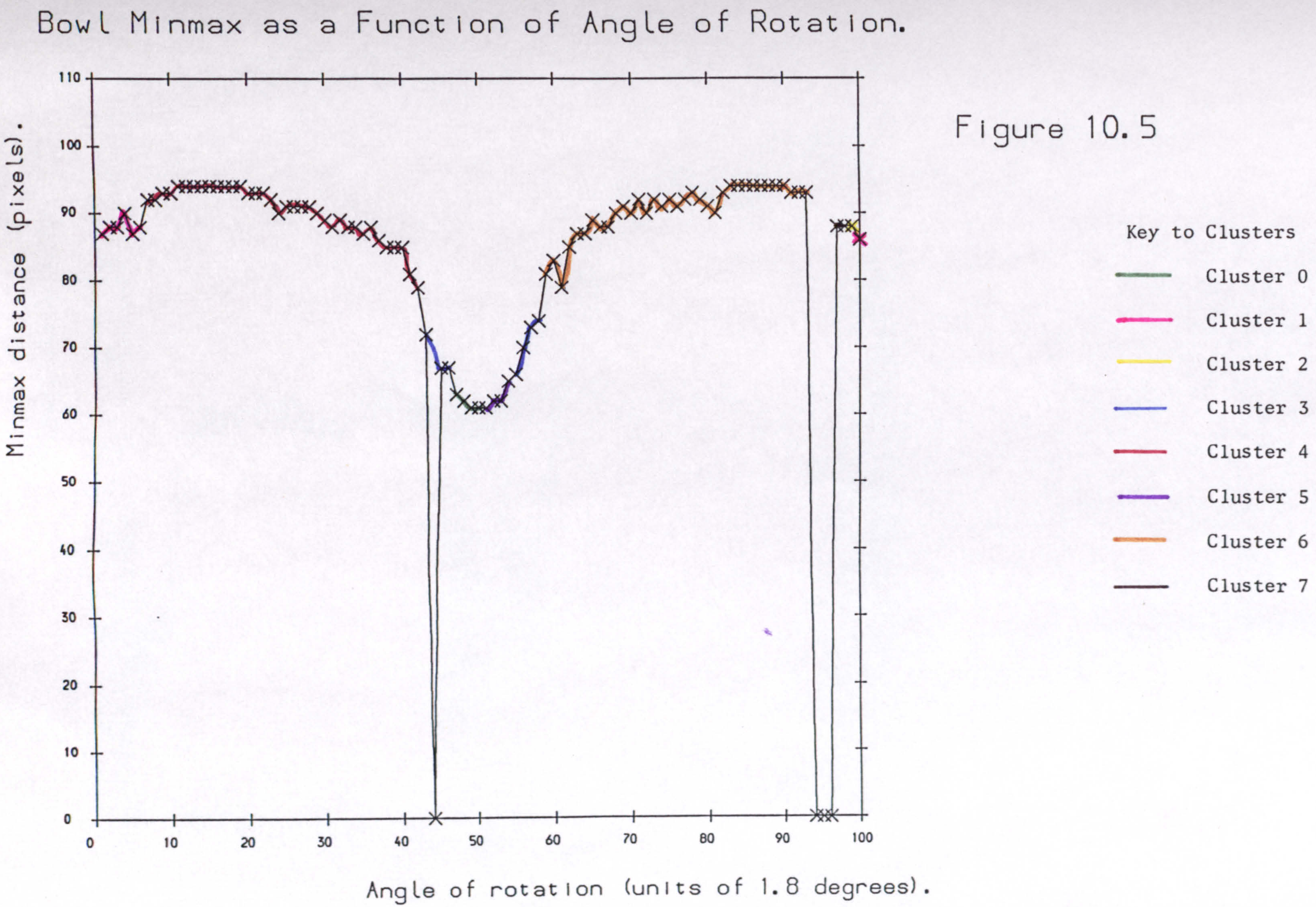

Bowl Minmax as a Function of Angle of Rotation.
Minmax distance (pixels).
110
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
Angle of rotation (units of 1.8 degrees).
Key to Clusters
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7

Figure 10.5

Bowl Minmid as a Function of Angle of Rotation.

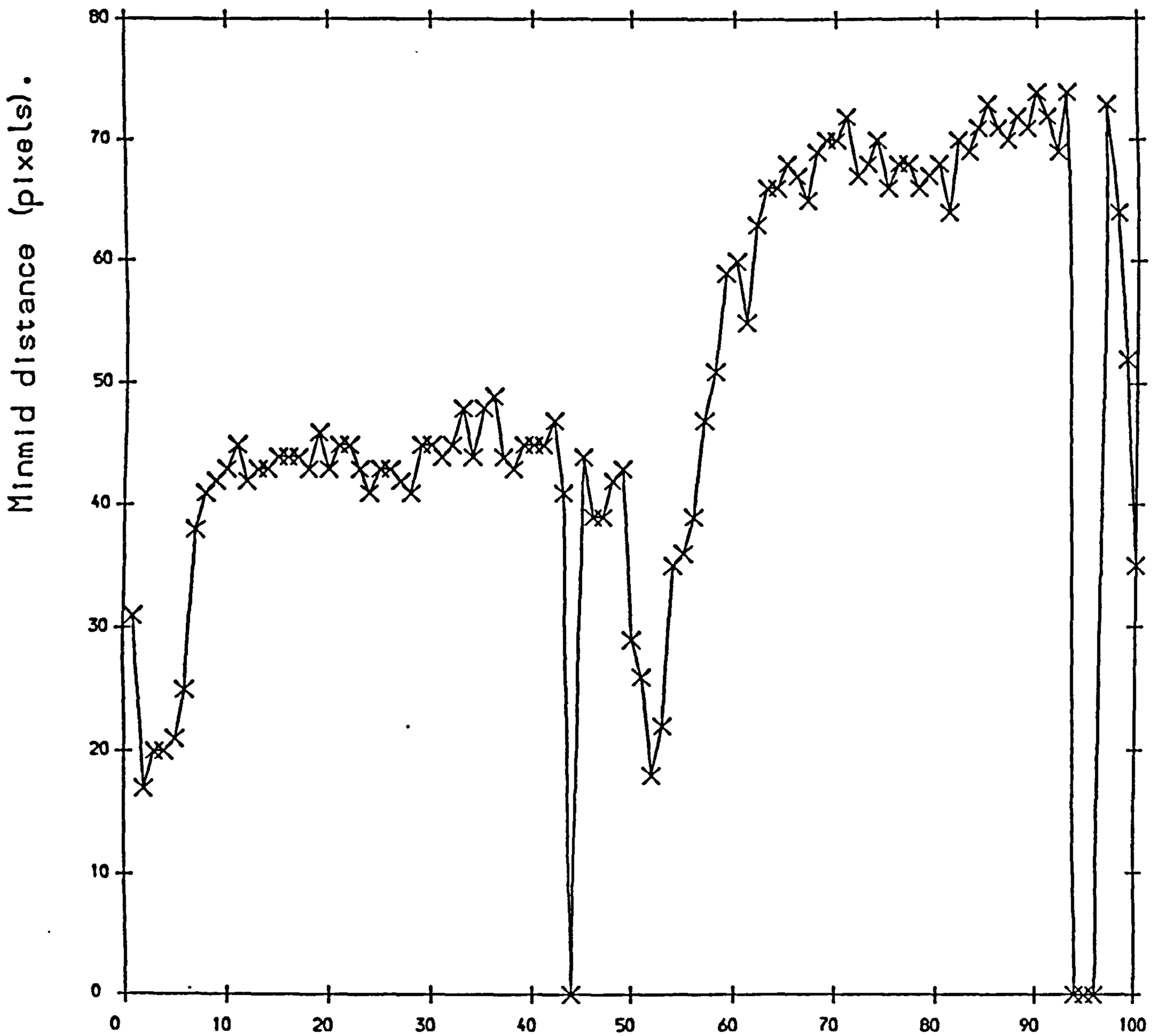


Figure 10.6

Bowl Midmax as a Function of Angle of Rotation.

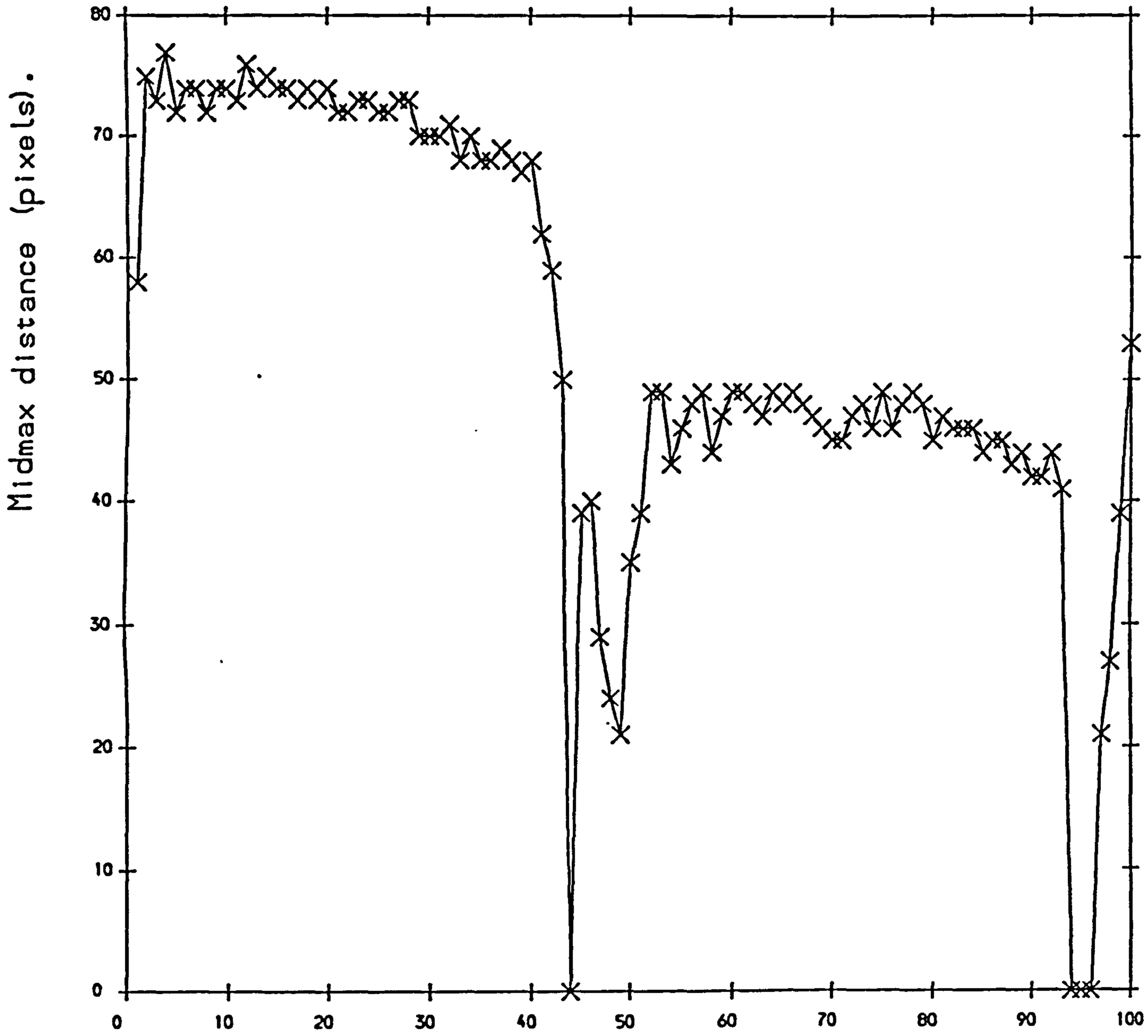


Figure 10.7

Cup Compaction as a Function of Angle of Rotation.

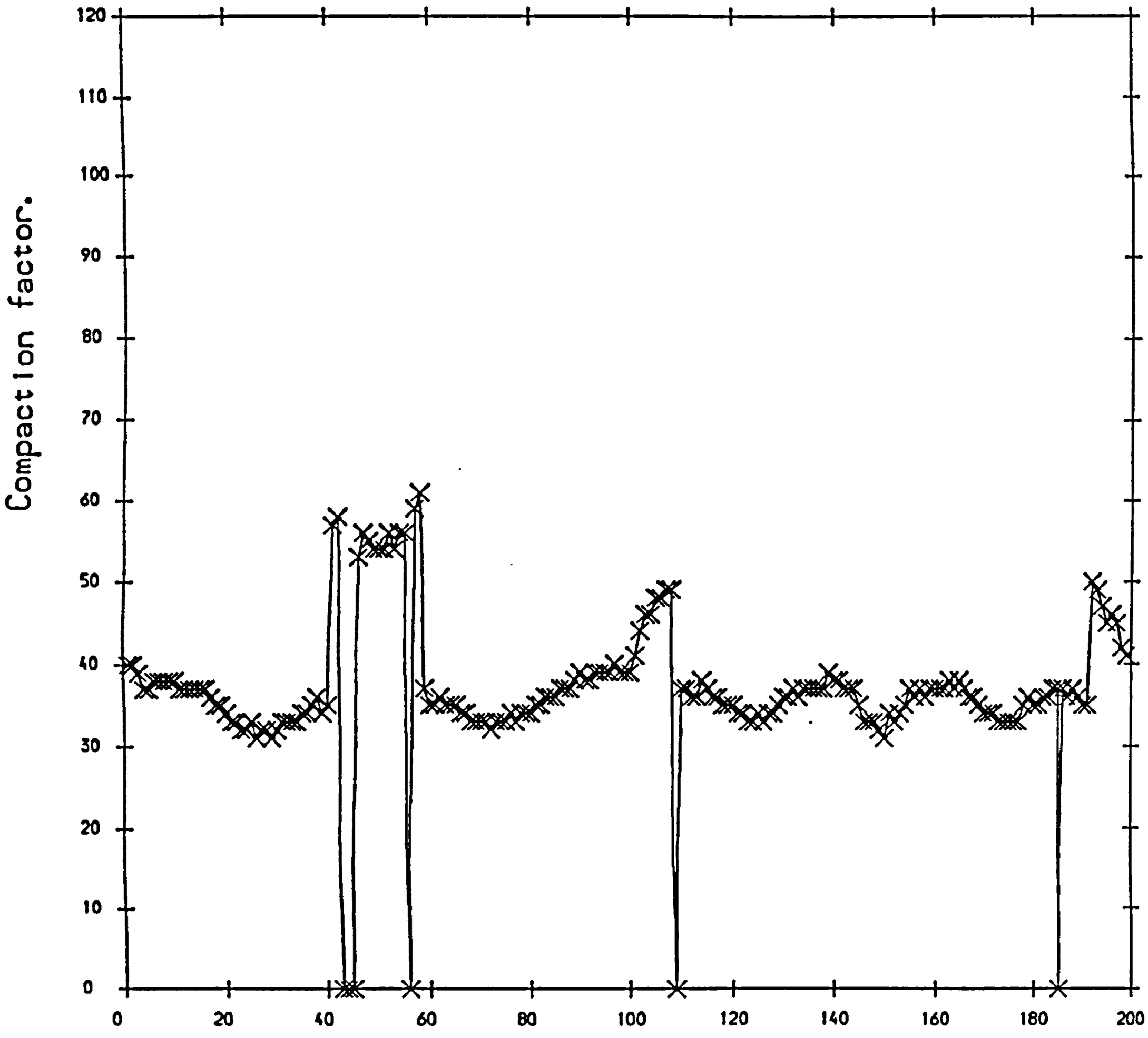


Figure 10.8

Cup Minmax as a Function of Angle of Rotation.

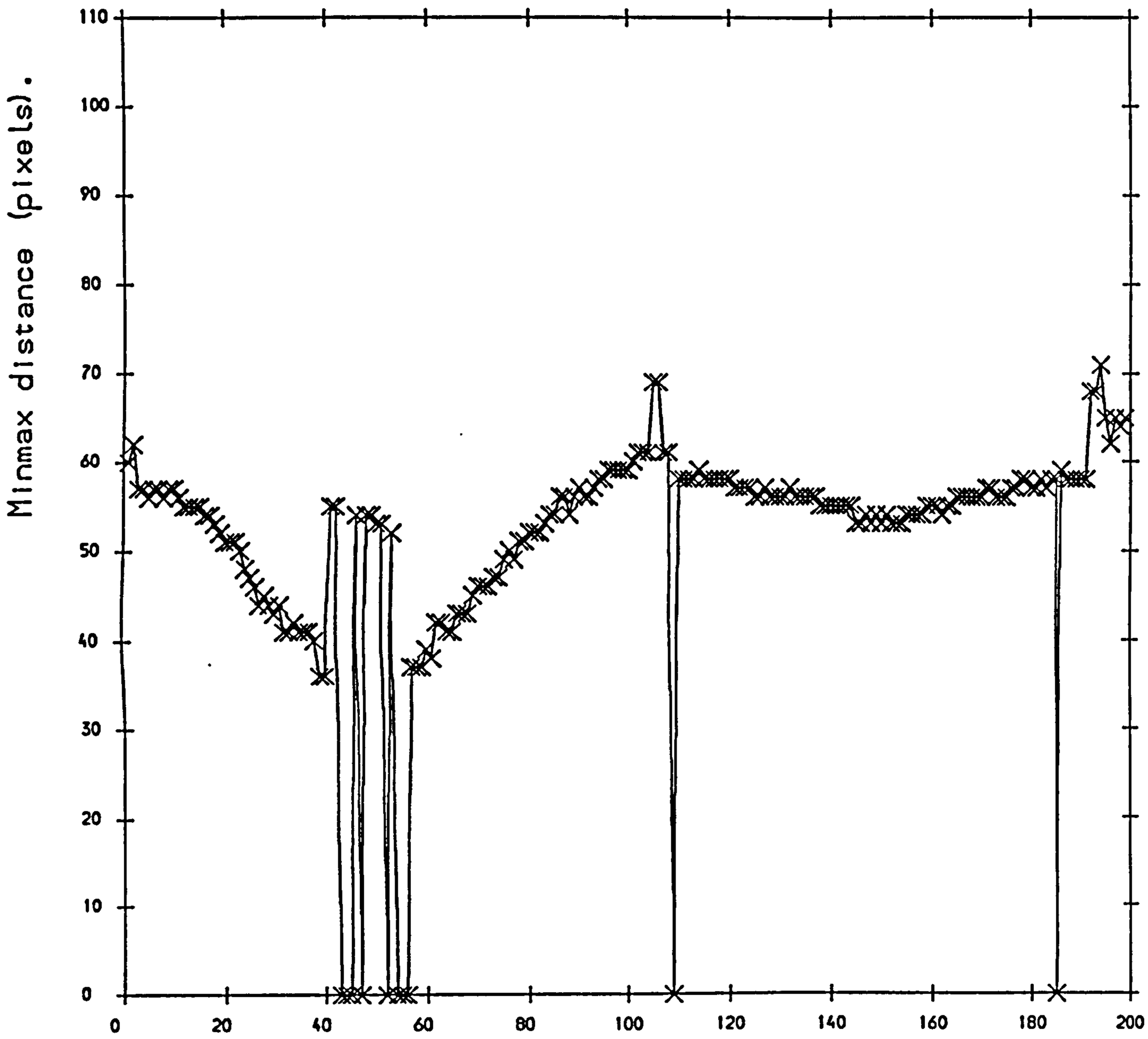


Figure 10.9

Cup Minmid as a Function of Angle of Rotation.

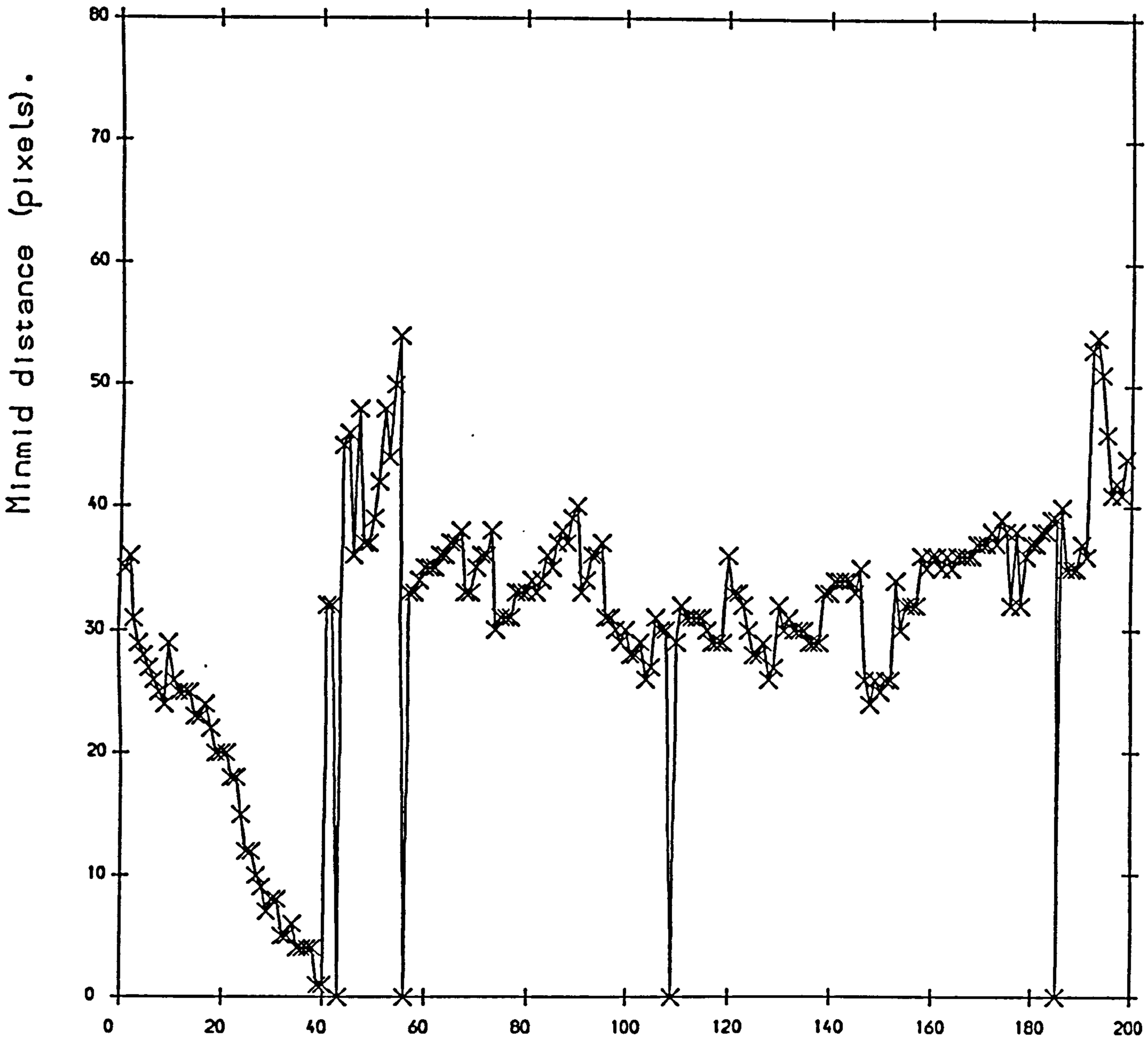


Figure 10.10

Cup Midmax as a Function of Angle of Rotation.

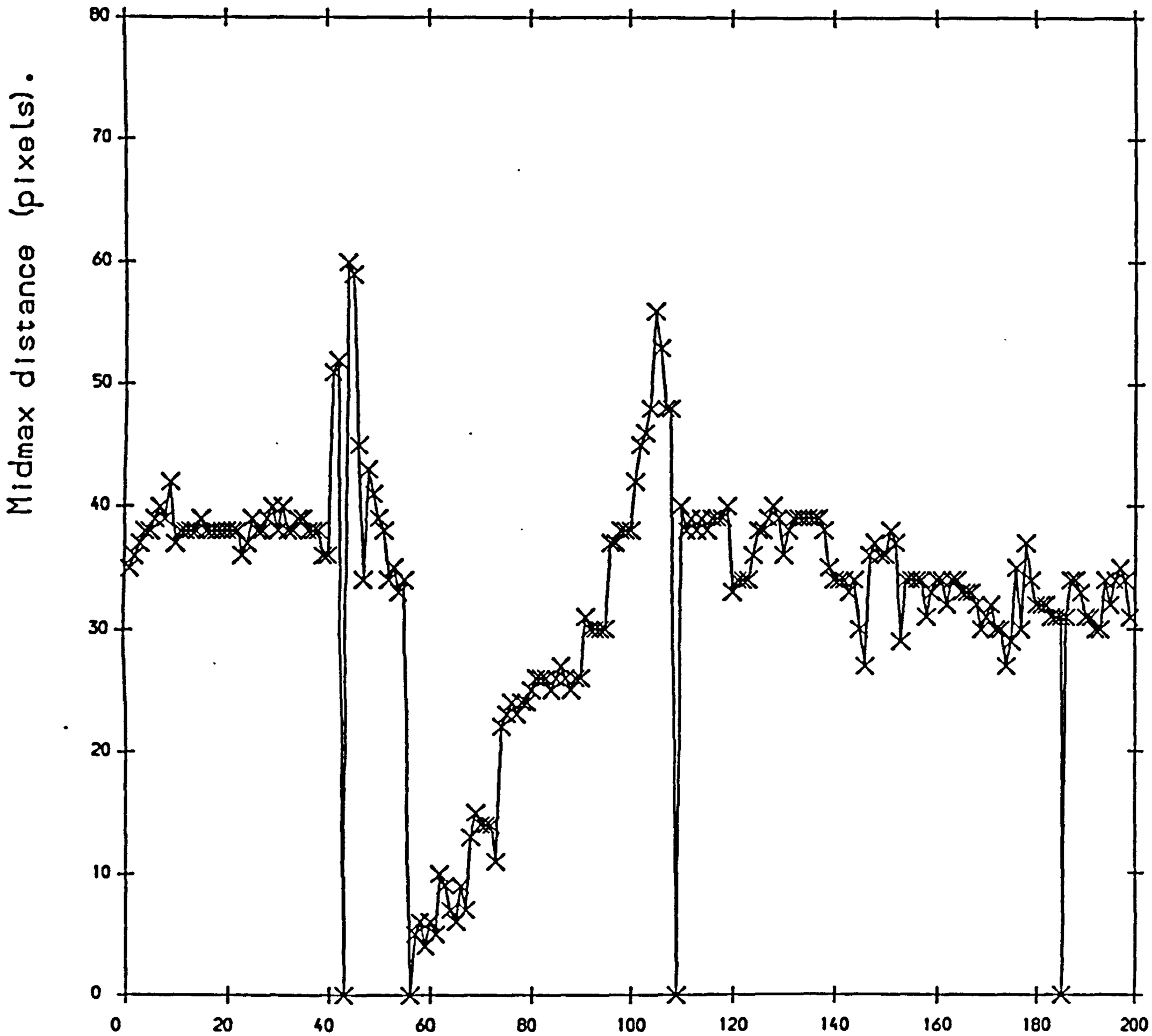


Figure 10.11

1	218	802	59	89	87	9	31	58	48	34	47
2	216	798	58	92	88	11	17	75	48	43	48
3	217	800	58	92	88	12	20	73	49	43	47
4	229	822	63	97	90	14	20	77	48	44	48
5	227	804	64	93	87	15	21	72	47	43	47
6	242	817	71	99	88	17	25	74	45	44	47
7	269	835	86	112	92	20	38	74	46	44	47
8	272	846	87	113	92	21	41	72	48	43	46
9	282	856	92	116	93	22	42	74	48	45	46
10	281	850	92	117	93	24	43	74	48	45	46
11	284	859	93	118	94	27	45	73	48	43	47
12	283	858	93	118	94	28	42	76	48	45	47
13	282	855	93	117	94	29	43	74	48	45	47
14	281	862	91	118	94	30	43	75	48	45	46
15	282	866	91	118	94	32	44	74	48	45	46
16	282	854	93	118	94	34	44	74	48	45	46
17	279	849	91	117	94	34	44	73	48	46	46
18	276	846	90	117	94	33	43	74	48	46	46
19	278	847	91	119	94	34	46	73	48	45	46
20	273	837	89	117	93	32	43	74	48	46	46
21	274	837	89	117	93	32	45	72	48	45	46
22	270	828	88	117	93	31	45	72	48	45	46
23	269	821	88	116	92	31	43	73	48	45	45
24	260	807	83	114	90	29	41	73	48	47	44
25	263	796	86	115	91	29	43	72	48	46	44
26	263	792	87	115	91	29	43	72	48	47	44
27	263	788	87	115	91	28	42	73	49	47	44
28	263	788	87	114	91	27	41	73	49	47	43
29	265	779	90	115	90	27	45	70	48	46	43
30	270	767	95	115	89	28	45	70	48	46	42
31	271	762	96	114	88	27	44	70	48	47	41
32	276	770	98	116	89	27	45	71	47	46	42
33	276	755	100	116	88	27	48	68	48	47	41
34	276	752	101	114	88	25	44	70	48	46	40
35	279	746	104	116	87	26	48	68	47	47	40
36	279	746	104	117	88	25	49	68	48	47	40
37	279	722	107	113	86	24	44	69	47	46	39
38	277	714	107	111	85	22	43	68	47	46	39
39	277	699	109	112	85	22	45	67	47	46	39
40	279	690	112	113	85	21	45	68	47	46	39
41	260	664	101	107	81	20	45	62	46	46	36
42	260	658	102	106	79	20	47	59	46	46	35
43	235	633	87	91	72	17	41	50	46	45	32
44	0	0	0	0	0	0	0	0	0	0	0
45	212	596	75	83	67	16	44	39	46	45	33
46	206	592	71	79	67	15	39	40	46	45	34
47	177	565	55	68	63	14	39	29	46	45	41
48	165	558	48	66	62	10	42	24	46	45	44
49	163	554	47	64	61	9	43	21	45	45	45
50	163	553	48	64	61	8	29	35	45	44	45
51	163	555	47	65	61	9	26	39	44	44	45
52	165	554	49	67	62	10	18	49	44	47	46
53	177	577	54	71	62	10	22	49	40	47	46
54	200	599	66	78	65	11	35	43	35	46	46
55	205	599	70	82	66	12	36	46	34	46	45
56	214	615	74	87	70	13	39	48	35	47	45
57	236	635	87	96	73	14	47	49	34	46	46
58	238	641	88	95	74	15	51	44	35	46	46
59	257	658	100	106	81	16	59	47	37	46	46
60	262	682	100	109	83	17	60	49	38	46	47
61	264	686	101	104	79	17	55	49	36	46	47
62	268	703	102	111	85	18	63	48	41	46	46
63	274	719	104	113	87	20	66	47	42	46	46
64	274	725	103	115	87	20	66	49	42	46	46
65	278	739	104	116	89	21	68	48	43	46	47
66	278	746	103	116	88	22	67	49	43	46	46
67	276	749	101	113	88	22	65	48	42	46	47
68	274	748	100	116	90	23	69	47	43	46	47
69	271	766	95	116	91	23	70	46	44	46	47
70	271	766	95	115	90	24	70	45	44	47	47
71	271	778	94	117	92	25	72	45	45	46	47
72	264	770	90	114	90	24	67	47	45	46	47
73	265	775	90	116	92	25	68	48	46	46	47
74	264	788	88	116	91	26	70	46	46	46	47
75	262	788	87	115	92	26	66	49	46	45	47
76	260	801	84	114	91	26	68	46	47	45	47
77	265	814	86	116	92	28	68	48	47	45	47
78	264	802	86	115	93	28	66	49	47	44	47
79	264	803	86	115	92	29	67	48	47	44	47
80	265	801	87	113	91	29	68	45	46	45	47
81	270	818	89	111	90	29	64	47	46	43	46
82	276	822	92	116	93	30	70	46	48	44	46
83	276	834	91	115	94	30	69	46	48	43	47
84	282	832	95	117	94	31	71	46	48	43	47
85	282	837	95	117	94	33	73	44	48	44	46
86	284	832	96	116	94	33	71	45	48	43	46
87	284	836	96	115	94	33	70	45	48	42	46
88	283	830	96	115	94	33	72	43	48	43	46
89	283	827	96	115	94	31	71	44	48	42	46
90	281	829	95	116	94	30	74	42	48	43	46
91	278	821	94	114	93	27	72	42	48	42	46
92	278	823	93	113	93	25	69	44	48	41	46
93	280	804	97	115	93	25	74	41	48	43	46
94	0	0	0	0	0	0	0	0	0	0	0
95	0	0	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0	0	0
97	228	767	67	94	88	13	73	21	48	42	46
98	221	777	62	91	88	10	64	27	49	37	48
99	217	770	61	91	88	9	52	39	49	33	48
100	213	777	58	88	86	8	35	53	47	33	48

Figure 10.12

Figure 10.13

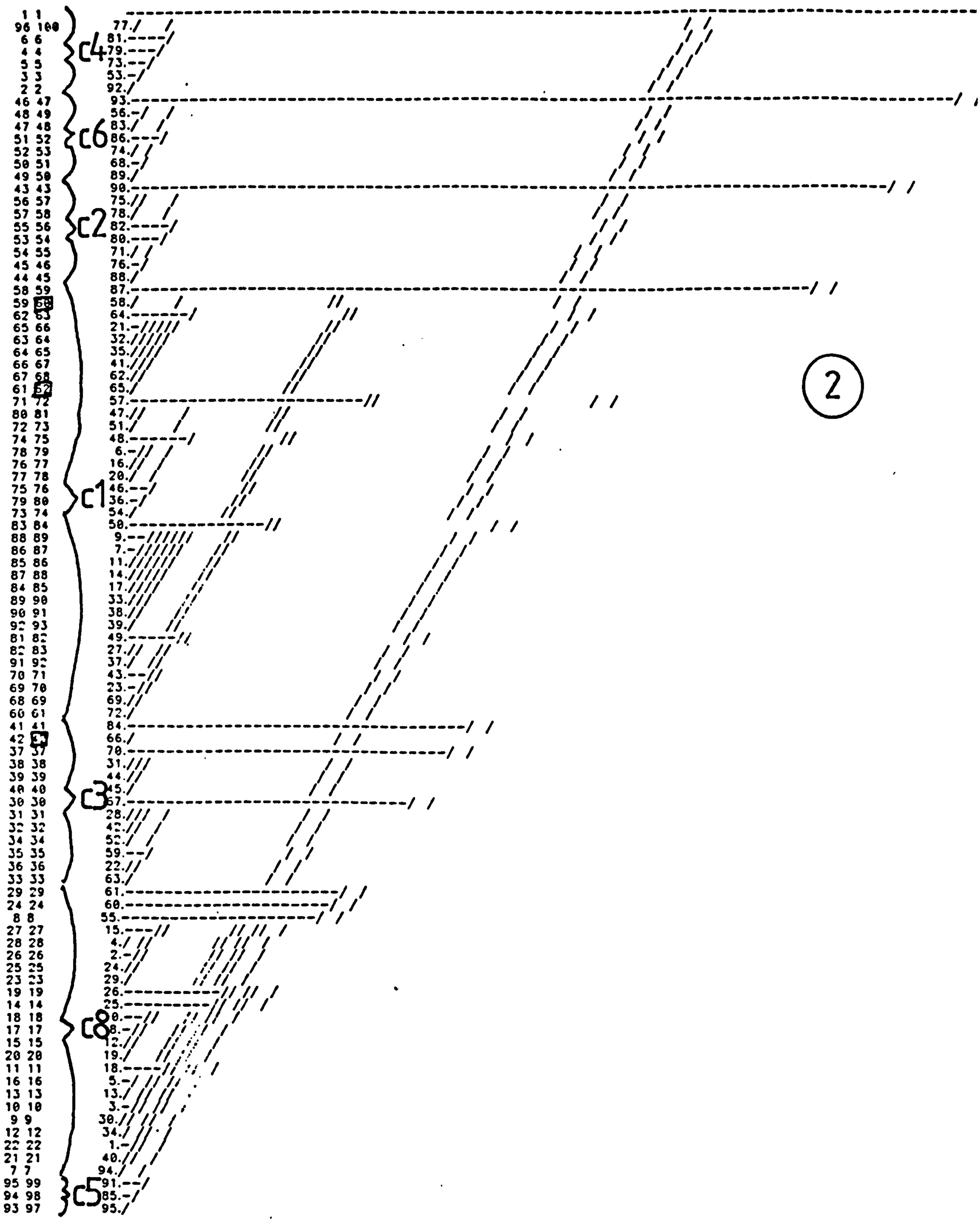
```
BMDP2M - SINGLE LINKAGE CLUSTER ANALYSIS OF CASES
BMDP STATISTICAL SOFTWARE, INC.
1964 WESTWOOD BLVD. SUITE 202
(213) 475-5700
PROGRAM REVISED OCTOBER 1983
MANUAL REVISED --- 1983
COPYRIGHT (C) 1983 REGENTS OF UNIVERSITY OF CALIFORNIA

TO SEE REMARKS AND A SUMMARY OF NEW FEATURES FOR
THIS PROGRAM, STATE NEWS. IN THE PRINT PARAGRAPH.

DEC 14, 1987        AT 15:37:42
```



```
PROGRAM CONTROL INFORMATION

/ PROBLEM           TITLE IS 'S-LINKAGE CLUSTER ANALYSIS OF BOWL FEATURE DATA'.

/ INPUT             VARIABLES ARE 12.
                    FORMAT IS FREE.
                    UNIT=7.
/ VARIABLE          NAMES ARE ID,PERIMETER,AREA,COMPACTION,TOTAL,MINMAX,
                    XENDMID,MINMID,MIDMAX,DtoMIN,DtoMID,DtoMAX.
                    USE=COMPACTION,MINMID,MIDMAX,MINMAX.
                    MINIMUM IS (4) 10.
                    LABEL IS ID.
/ PROCEDURE         LINK IS SINGLE.
/ PRINT             HORIZ.
                    NCUT=5.
/ END

PROBLEM TITLE IS
S-LINKAGE CLUSTER ANALYSIS OF BOWL FEATURE DATA

NUMBER OF VARIABLES TO READ IN. . . . . .      12
NUMBER OF VARIABLES ADDED BY TRANSFORMATIONS. .      0
TOTAL NUMBER OF VARIABLES . . . . . .      12
NUMBER OF CASES TO READ IN. . . . . . TO END
CASE LABELING VARIABLES . . . . . .ID
MISSING VALUES CHECKED BEFORE OR AFTER TRANS. .  BEFORE
BLANKS ARE. . . . . . MISSING
INPUT UNIT NUMBER . . . . . .      7
REWIND INPUT UNIT PRIOR TO READING. . DATA. . .    YES
NUMBER OF WORDS OF DYNAMIC STORAGE. . . . . .  14998

VARIABLES TO BE USED
     4 COMPACTI    8 MINMID      9 MIDMAX      6 MINMAX
INPUT FORMAT IS
FREE

MAXIMUM LENGTH DATA RECORD IS   80 CHARACTERS.

NUMBER OF CASES READ. . . . . .      96
PRINT DISTANCE MATRIX . . . . . .      NO
TYPE OF TREE PRINTED. . . . . .HORIZONTAL
CALCULATING PROCEDURE . . . . . SUM-SOR
STANDARDIZATION OF INPUT DATA . . . . . .     YES
AMALGAMATION RULE . . . . . SINGLE
NUMBER OF NEIGHBORS USED FOR DISTANCE CALC. . .      1
```

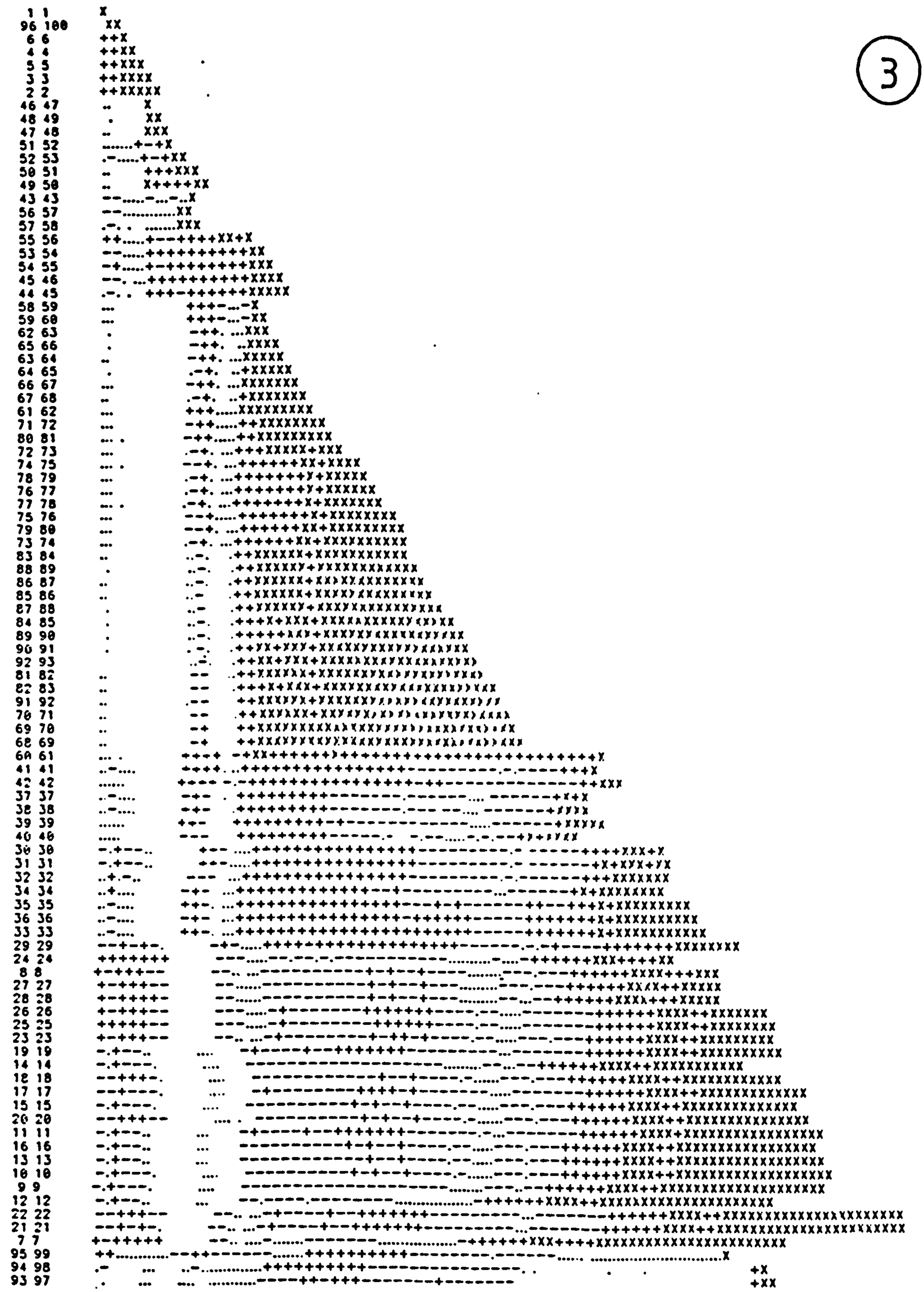
BMDP3M – SINGLE LINKAGE CLUSTER ANALYSIS OF CASES CONTINUED

Figure 10.13 Continued.



DISTANCES BETWEEN CASES REPRESENTED IN SHADED FORM.
HEAVY SHADING INDICATES SMALL DISTANCES.

Figure 10.13 Continued.



THE DISTANCES HAVE BEEN REPRESENTED ABOVE IN SHADED
FORM ACCORDING TO THE FOLLOWING SCHEME

X		LESS THAN		1.022
+	FROM	1.022	TO	2.295
-	FROM	2.295	TO	2.661
.	FROM	2.661	TO	3.644
		GREATER THAN		3.644

Figure 10.14

(1)

```
BMDPKM - K MEANS CLUSTERING OF CASES
BMDP STATISTICAL SOFTWARE, INC.
1964 WESTWOOD BLVD. SUITE 202
(213) 475-5700
COPYRIGHT (C) 1983 REGENTS OF UNIVERSITY OF CALIFORNIA

DEC 14, 1987        AT 15:38:47

PROGRAM CONTROL INFORMATION

/ PROBLEM           TITLE IS  'K-CLUSTER ANALYSIS OF BOWL FEATURE DATA'.

/ INPUT             VARIABLES ARE 12.
                    FORMAT IS FREE.
                    UNIT=7.
/ VARIABLE          NAMES ARE ID,PERIMETER,AREA,COMPACTION,TOTAL,MINMAX,
                    XENDMID,MINMID,MIDMAX,DtoMin,DtoMid,DtoMax.
                    LABEL IS ID.
                    MINIMUM IS (4) 10.
                    USE=COMPACTION,MINMID,MIDMAX,MINMAX.
/ CLUSTER           NUMBER IS 8.
                    STANDARDIZE IS VAR.
/ PRINT             MEMB.
/ PLOT              SIZE IS 60,35.
                    XVAR IS ID.
                    YVAR IS MINMAX.
/ END

PROBLEM TITLE IS
K-CLUSTER ANALYSIS OF BOWL FEATURE DATA

NUMBER OF VARIABLES TO READ IN. . . . . .      12
NUMBER OF VARIABLES ADDED BY TRANSFORMATIONS. .       0
TOTAL NUMBER OF VARIABLES . . . . . .      12
NUMBER OF CASES TO READ IN. . . . . . TO END
CASE LABELING VARIABLES . . . . . .ID
MISSING VALUES CHECKED BEFORE OR AFTER TRANS. .  BEFORE
BLANKS ARE. . . . . . MISSING
INPUT UNIT NUMBER . . . . . .       7
REWIND INPUT UNIT PRIOR TO READING. . DATA. . .     YES
NUMBER OF WORDS OF DYNAMIC STORAGE. . . . . .   14998

VARIABLES TO BE USED
     4 COMPACTI     6 MINMID      9 MIDMAX      6 MINMAX

MAXIMUM LENGTH DATA RECORD IS   80 CHARACTERS.

  NUMBER(S) OF CLUSTER(S) TO REPORT . . . . . .      8
  DISTANCES ARE STANDARDIZED BY . . . . . VAR
  MAXIMUM NUMBER OF MISSING VALUES PER CASE . . .     2
  TOLERANCE . . . . . 0.001000
  MAXIMUM NUMBER OF ITERATIONS. . . . . .    30
  MAXIMUM NUMBER OF CASES THAT CAN BE PROCESSED    1169

NUMBER OF CASES READ. . . . . .      96
```

Figure 10.14 Continued.

BMDPKM – K MEANS CLUSTERING OF CASES CONTINUED

REALLOCATION IS COMPLETE AFTER 6 ITERATIONS FOR 8 CLUSTERS

CLUSTER 1 OF 8 CONTAINS 31 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
                1
              1  1
            1 1  1
            1111 1
            111111 1  1
          1 11111111111
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                3.0000                6.0000
```

2a

```
                                        888
                                        888
                                        588
                                       8588
                                     78 85888
                                     383833888 8     4       4       4
                      7  7  7    7    33233323388 2     22  4 2  2  6 4   4  64   6  666 6 6
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS              3.0000                6.0000
```

2b

CASE	WEIGHT	DISTANCE	I	VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
63	1.0000	0.8259	I	4 COMPACTI	5.2623	5.8767	6.5153	0.3567
64	1.0000	0.8018	I	8 MINMID	4.1500	4.4784	4.7985	0.1690
65	1.0000	0.7107	I	9 MIDMAX	2.7596	3.0939	3.2980	0.1469
66	1.0000	0.7297	I	6 MINMAX	9.0458	9.5254	9.7737	0.2273
67	1.0000	0.6575	I					
68	1.0000	0.4284	I					
69	1.0000	0.1155	I					
70	1.0000	0.2040	I					
71	1.0000	0.2055	I					
72	1.0000	0.3282	I					
73	1.0000	0.2863	I					
74	1.0000	0.3742	I					
75	1.0000	0.5144	I					
76	1.0000	0.6215	I					
77	1.0000	0.5141	I					
78	1.0000	0.5840	I					
79	1.0000	0.5267	I					
80	1.0000	0.4414	I					
81	1.0000	0.4811	I					
82	1.0000	0.1931	I					
83	1.0000	0.3042	I					
84	1.0000	0.2880	I					
85	1.0000	0.3871	I					
86	1.0000	0.3170	I					
87	1.0000	0.2974	I					
88	1.0000	0.3958	I					
89	1.0000	0.3373	I					
90	1.0000	0.4909	I					
91	1.0000	0.3585	I					
92	1.0000	0.2023	I					
93	1.0000	0.5245	I					

3

4

AVERAGE DISTANCE 0.4338

CLUSTER 2 OF 8 CONTAINS 7 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
                    2
          2  2 2 2 2  2
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                1.7500                3.5000
```

```
                                                          8
                                                         88
                                                       8 83          8
                                                      83 33  8       88
                                                      81 33  4      888
                                                      31511  3 8    854
                                        7     7     6 7 6   6113118 1811 881118  4
                          6           7  6  7 3 64 5 6 4 6 1  131111141181188111111 4
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS              1.7500                3.5000
```

CASE	WEIGHT	DISTANCE	I	VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
43	1.0000	0.6522	I	4 COMPACTI	4.3853	4.9401	5.5129	0.5082
45	1.0000	0.5727	I	8 MINMID	2.3344	2.7512	3.3071	0.3387
46	1.0000	0.7069	I	9 MIDMAX	2.6250	3.0384	3.3654	0.2919
55	1.0000	0.8036	I	6 MINMAX	6.8624	7.2634	7.6942	0.3365
56	1.0000	0.4233	I					
57	1.0000	0.7227	I					
58	1.0000	0.9102	I					

AVERAGE DISTANCE 0.6845

CLUSTER 3 OF 8 CONTAINS 11 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
          3
          33  3
         3333 33 3
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                3.0000                6.0000
```

```
                        8
                        88                1
```

```
                    88                1
                    888               1
                    888           2 2 14
                    888           2 1 11 1          5
                    888  8  7 7 1   1111111 1 1    4   4 4                    6
                 8   87 888  8   7 71111 1111111 1 1 2444  222  6   5   5 6  66  6  6 6
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS                  3.0000                    6.0000
```

CASE	WEIGHT	DISTANCE		VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
34	1.0000	0.2825	4	COMPACTI	6.0141	6.4868	7.0164	0.3061
37	1.0000	0.2468	8	MINMID	2.7883	2.9475	3.1774	0.1276
38	1.0000	0.3038	9	MIDMAX	4.1730	4.5830	4.7788	0.1577
39	1.0000	0.3783	6	MINMAX	8.4220	8.9797	9.2538	0.2383
40	1.0000	0.5491						
35	1.0000	0.1802						
36	1.0000	0.2875						
33	1.0000	0.3249						
32	1.0000	0.4848						
31	1.0000	0.5271						
41	1.0000	0.7109						

AVERAGE DISTANCE 0.3887

CLUSTER 4 OF 8 CONTAINS 7 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
                    4
                    4
                 4 4   4  4       4
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                2.5000                    5.0000
```

```
                                                    7
                                                    6                1
                                                    3    3           1
                                  8    8 888          3    1 6 7   76 1 11
                                  88   888888     3 6 3  3  1 1 17  51 1 11 1    6
                              8 8 88 8 8888883 23532232632  22 1117116111111111111611  5
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS               2.5000                    5.0000
```

CASE	WEIGHT	DISTANCE		VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
3	1.0000	0.4482	4	COMPACTI	3.6335	3.8572	4.4479	0.3045
4	1.0000	0.6611	8	MINMID	1.1024	1.5655	2.2695	0.4276
5	1.0000	0.3390	9	MIDMAX	3.5673	4.6346	5.1826	0.6307
6	1.0000	0.6875	6	MINMAX	8.9419	9.1201	9.3578	0.1303
100	1.0000	1.3102						
1	1.0000	0.8736						
2	1.0000	0.6606						

AVERAGE DISTANCE 0.7115

CLUSTER 5 OF 8 CONTAINS 3 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
                 5          5 5
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                2.5000                    5.0000
```

```
                                                    1
                                                    1      2           8   888
                                                    1      2           8 8 888
                                  1   1   14    7 1         .          888 838   3
                                 111  1 1112   27 14     6    8 8838838  836   6
                             1 1 111 1111111  11271172 626  6 78383863364434  344
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS               2.5000                    5.0000
```

CASE	WEIGHT	DISTANCE		VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
97	1.0000	0.8736	4	COMPACTI	3.8214	3.9676	4.1973	0.2014
98	1.0000	0.1712	8	MINMID	3.3719	4.0852	4.7336	0.6832
99	1.0000	0.9915	9	MIDMAX	1.4134	1.9519	2.6250	0.6169
			6	MINMAX	9.1498	9.1498	9.1498	0.0000

AVERAGE DISTANCE 0.6788

CLUSTER 6 OF 8 CONTAINS 8 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

```
                       6
                 6  6  6     66 66
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN THIS CLUSTER                3.0000                    6.0000
```

```
                                                                8    8
                                                                8    8
                                                                8    88
                                                               88   88388 1
                                                               87   83388 1
                                                               13   3313881
                                                             7    8 11  3113831
                                  :                        4  5     7 7    1 11881111111
                                  :    2 2          2 2 2 4      5  44 44 5437 1 11811111111
DISTANCE +.....+.....+.....+.....+.....+.....+.....+.....+.....+.....+
FROM CENTER TO CASES IN OTHER CLUSTERS               3.0000                    6.0000
```

CASE	WEIGHT	DISTANCE		VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.

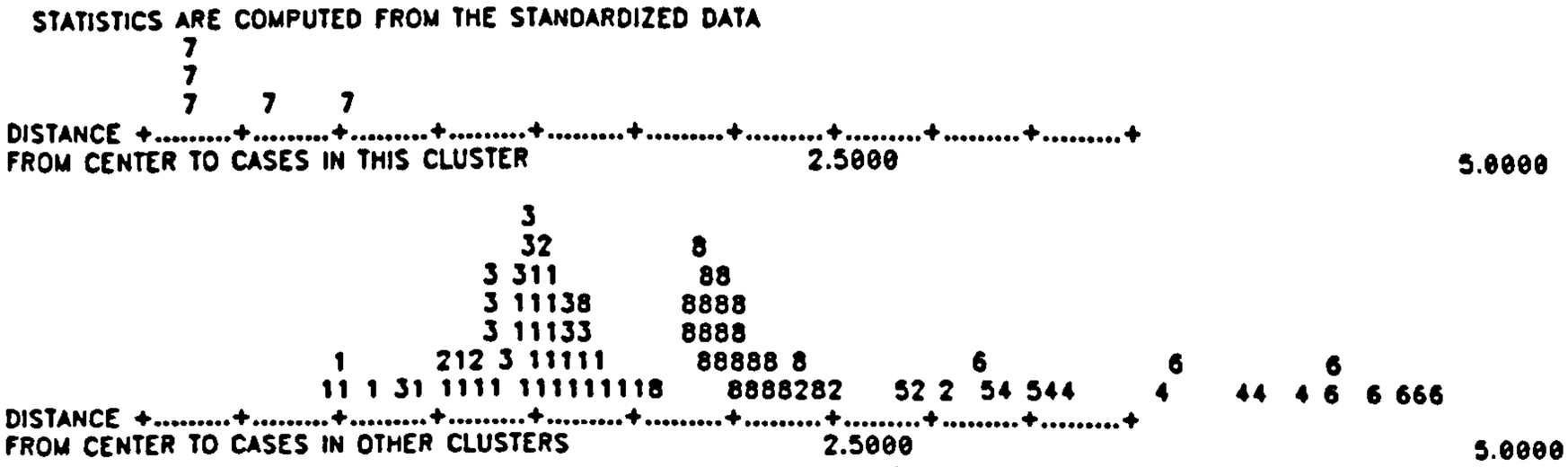
Figure 10.14 Continued.

CASE	WEIGHT	DISTANCE	VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
47	1.0000	0.7076	4 COMPACTI	2.9444	3.2420	4.1347	0.4098
48	1.0000	1.0785	8 MINMID	1.1672	2.0588	2.7883	0.6100
49	1.0000	1.2926	9 MIDMAX	1.4134	2.4315	3.2980	0.7262
50	1.0000	0.3262	6 MINMAX	6.3425	6.4594	6.7584	0.1410
51	1.0000	0.5279					
52	1.0000	1.2553					
53	1.0000	1.0820					
54	1.0000	1.0700					

AVERAGE DISTANCE 0.9175

CLUSTER 7 OF 8 CONTAINS 5 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA

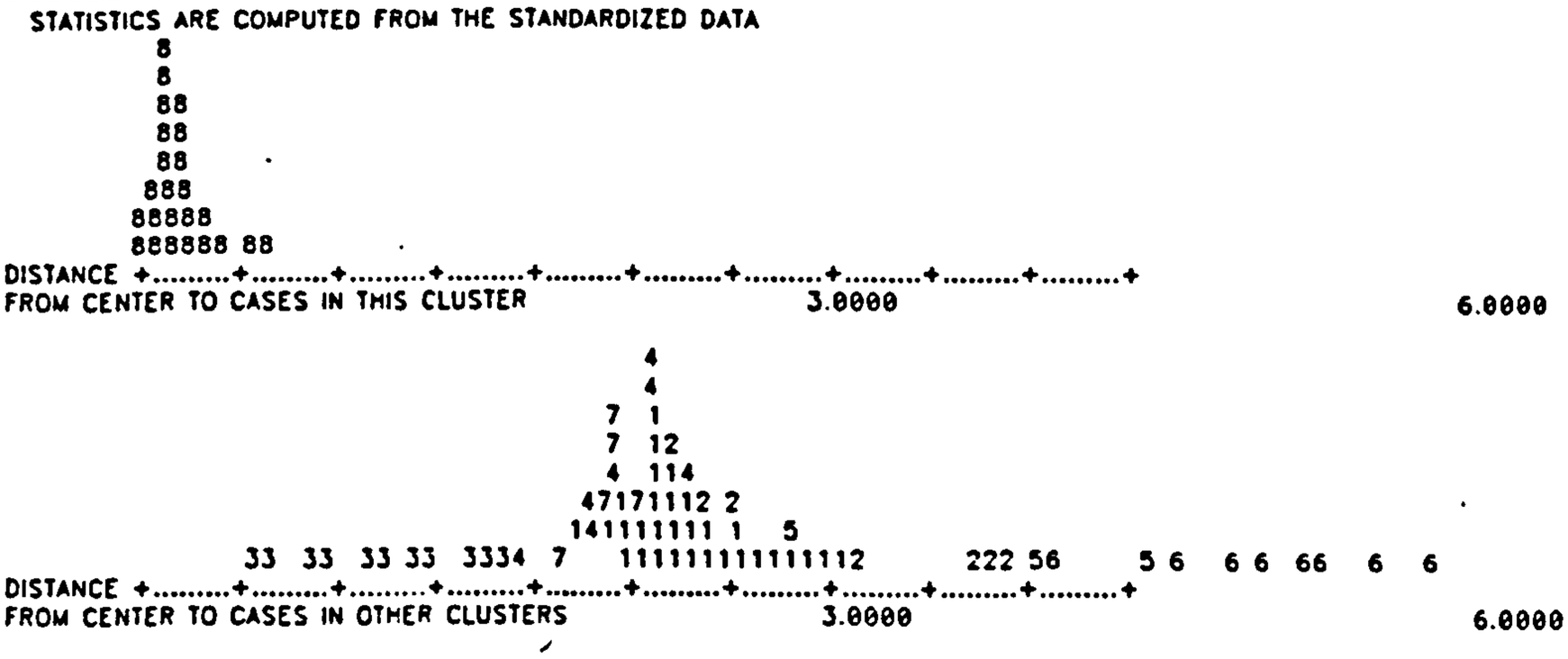


CASE	WEIGHT	DISTANCE	VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
42	1.0000	0.8973	4 COMPACTI	6.2647	6.3273	6.3900	0.0626
59	1.0000	0.2800	8 MINMID	3.0477	3.6832	4.0852	0.4008
61	1.0000	0.2912	9 MIDMAX	3.1634	3.3923	3.9711	0.3284
60	1.0000	0.2890	6 MINMAX	8.2140	8.4636	8.8379	0.2711
62	1.0000	0.5760					

AVERAGE DISTANCE 0.4667

CLUSTER 8 OF 8 CONTAINS 24 CASES

STATISTICS ARE COMPUTED FROM THE STANDARDIZED DATA



CASE	WEIGHT	DISTANCE	VARIABLE	MINIMUM	CENTER	MAXIMUM	ST.DEV.
8	1.0000	0.2327	4 COMPACTI	5.1997	5.6173	5.9514	0.1835
9	1.0000	0.1811	8 MINMID	2.4641	2.7937	2.9828	0.1146
10	1.0000	0.1671	9 MIDMAX	4.7115	4.9190	5.1153	0.0929
11	1.0000	0.2888	6 MINMAX	9.2538	9.6177	9.7737	0.1622
12	1.0000	0.3339					
13	1.0000	0.2679					
14	1.0000	0.2190					
15	1.0000	0.1965					
16	1.0000	0.2744					
17	1.0000	0.1867					
18	1.0000	0.1691					
20	1.0000	0.0910					
21	1.0000	0.1588					
22	1.0000	0.1854					
23	1.0000	0.1169					
25	1.0000	0.2871					
26	1.0000	0.2399					
27	1.0000	0.2392					
28	1.0000	0.2655					
29	1.0000	0.3557					
30	1.0000	0.5501					
7	1.0000	0.4098					
24	1.0000	0.5102					
19	1.0000	0.2591					

AVERAGE DISTANCE 0.2577

REPORT ON CASES WITH POSITIVE WEIGHT

CLUSTER MEANS

	COMPACTI	MINMID	MIDMAX	MINMAX
1	93.8063	69.0644	45.9677	91.6128
2	78.8571	42.4286	45.1428	69.8571
3	103.5454	45.4545	68.0909	86.3636

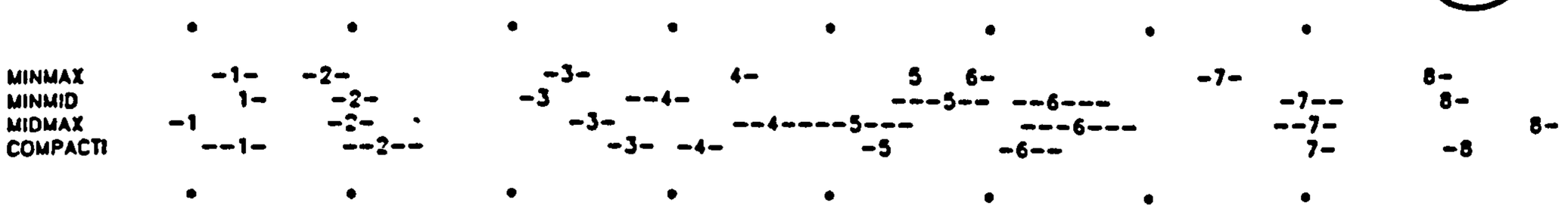
4	61.5714	24.1428	68.8571	87.7143
5	63.3333	63.0000	29.0000	88.0000
6	51.7500	31.7500	36.1250	62.1250
7	101.0000	56.8000	50.4000	81.4000
8	89.6666	43.0833	73.0832	92.4999

GRAND MEAN 86.3643 50.7080 55.7705 86.2601

CLUSTER STANDARD DEVIATIONS

	COMPACTI	MINMID	MIDMAX	MINMAX
1	5.6945	2.6069	2.1830	2.1860
2	8.1123	5.2236	4.3370	3.2367
3	4.8860	1.9679	2.3432	2.2923
4	4.8600	6.5937	9.3707	1.2536
5	3.2145	10.5357	9.1652	0.0
6	6.5411	9.4074	10.7894	1.3562
7	1.0000	6.1806	4.8785	2.6077
8	2.9291	1.7673	1.3805	1.5603
MEAN SQUARES				
BETWEEN	3167.2192	3011.2617	2744.6985	1215.4453
WITHIN	25.7912	19.6967	22.2669	4.1394
D.F.-S	7, 88	7, 88	7, 88	7, 88
F-RATIO	122.802	152.882	123.263	293.632
P-VALUE	0.0	0.0	0.0	0.0

CLUSTER PROFILES - VARIABLES ARE ORDERED BY F-RATIO SIZE



EACH COLUMN DESCRIBES A CLUSTER .
THE CLUSTER NUMBER IS PRINTED AT THE MEAN OF EACH VARIABLE
DASHES INDICATE ONE STANDARD DEVIATION ABOVE AND BELOW

POOLED WITHIN CLUSTER COVARIANCES

		COMPACTI 4	MINMID 8	MIDMAX 9	MINMAX 6
COMPACTI	4	25.79			
MINMID	8	4.11	19.70		
MIDMAX	9	3.76	-16.65	22.27	
MINMAX	6	0.69	2.86	1.16	4.14

POOLED WITHIN CLUSTER CORRELATIONS

		COMPACTI 4	MINMID 8	MIDMAX 9	MINMAX 6
COMPACTI	4	1.0000			
MINMID	8	0.1824	1.0000		
MIDMAX	9	0.1569	-0.7950	1.0000	
MINMAX	6	0.0668	0.3163	0.1209	1.0000

CPU TIME USED 0.757 SECONDS

BMDPKM - K MEANS CLUSTERING OF CASES
DEC 14, 1987 AT 15:38:56
PROGRAM CONTROL INFORMATION
NO MORE CONTROL LANGUAGE.
PROGRAM TERMINATED

Figure 10.14 Continued.

Schematic Diagram Illustrating the Need for Cluster Distinctions Based on Features Other Than Pick–up Point Distance Measurements.

(The plots are of the anticipated features for a rotating cube).

Figure 10.15a

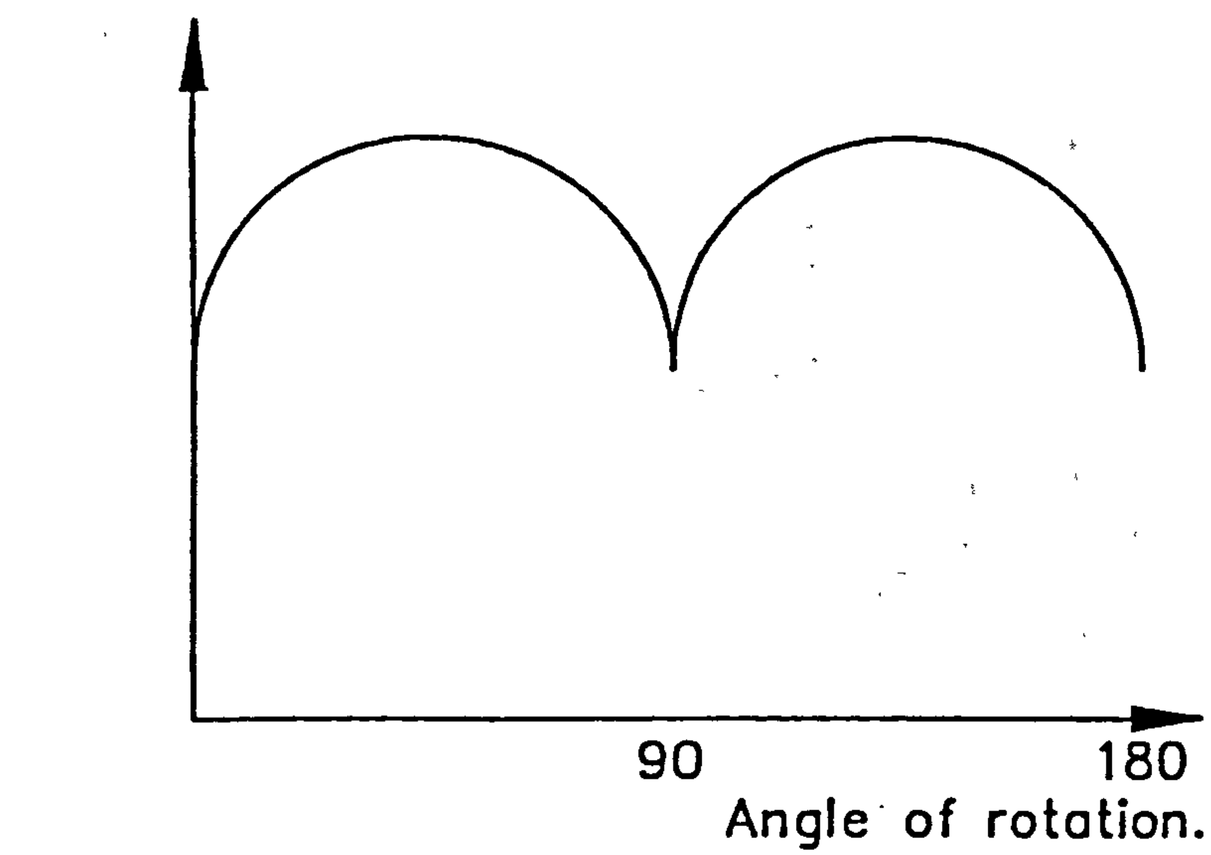


Figure 10.15b

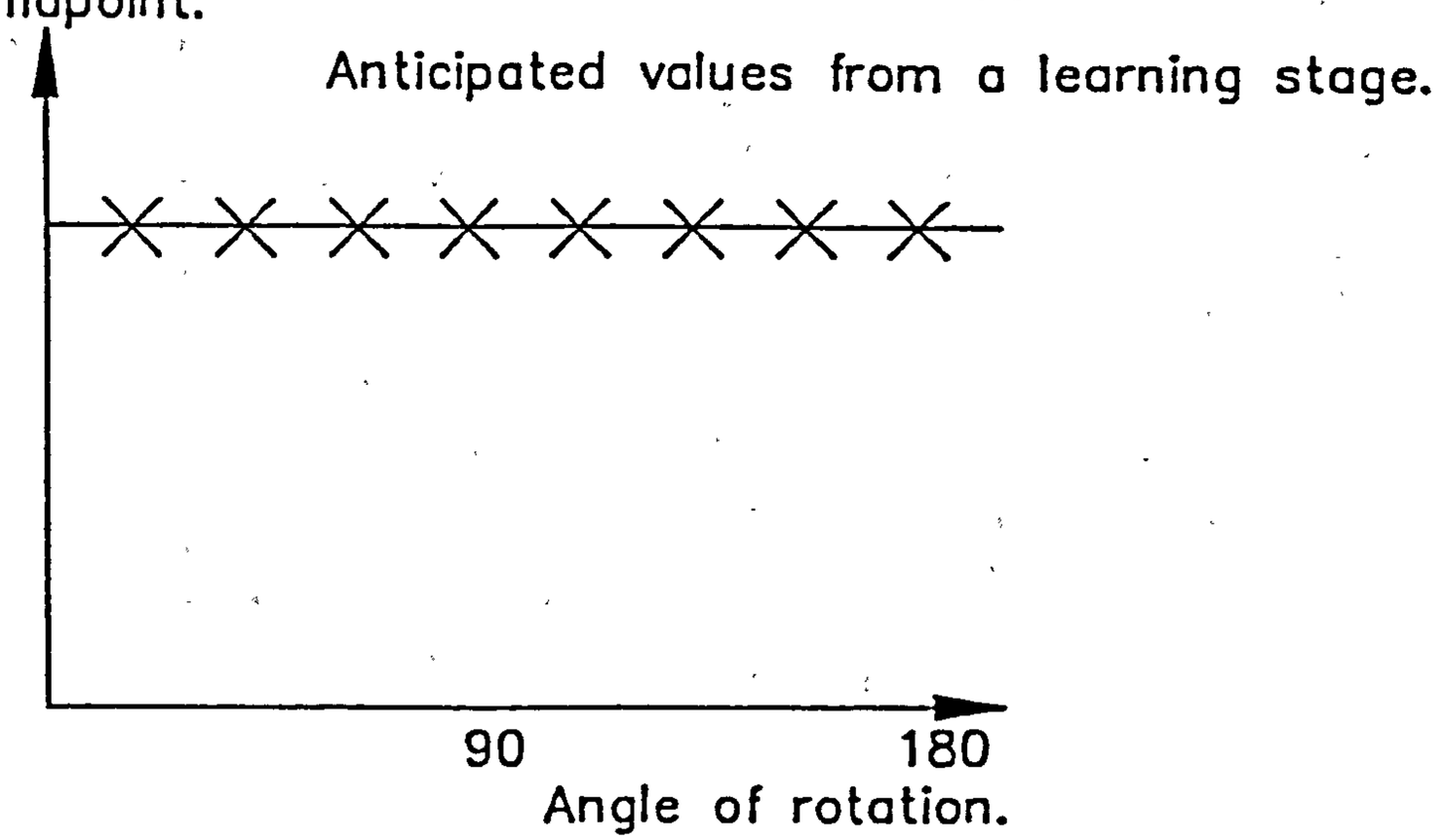
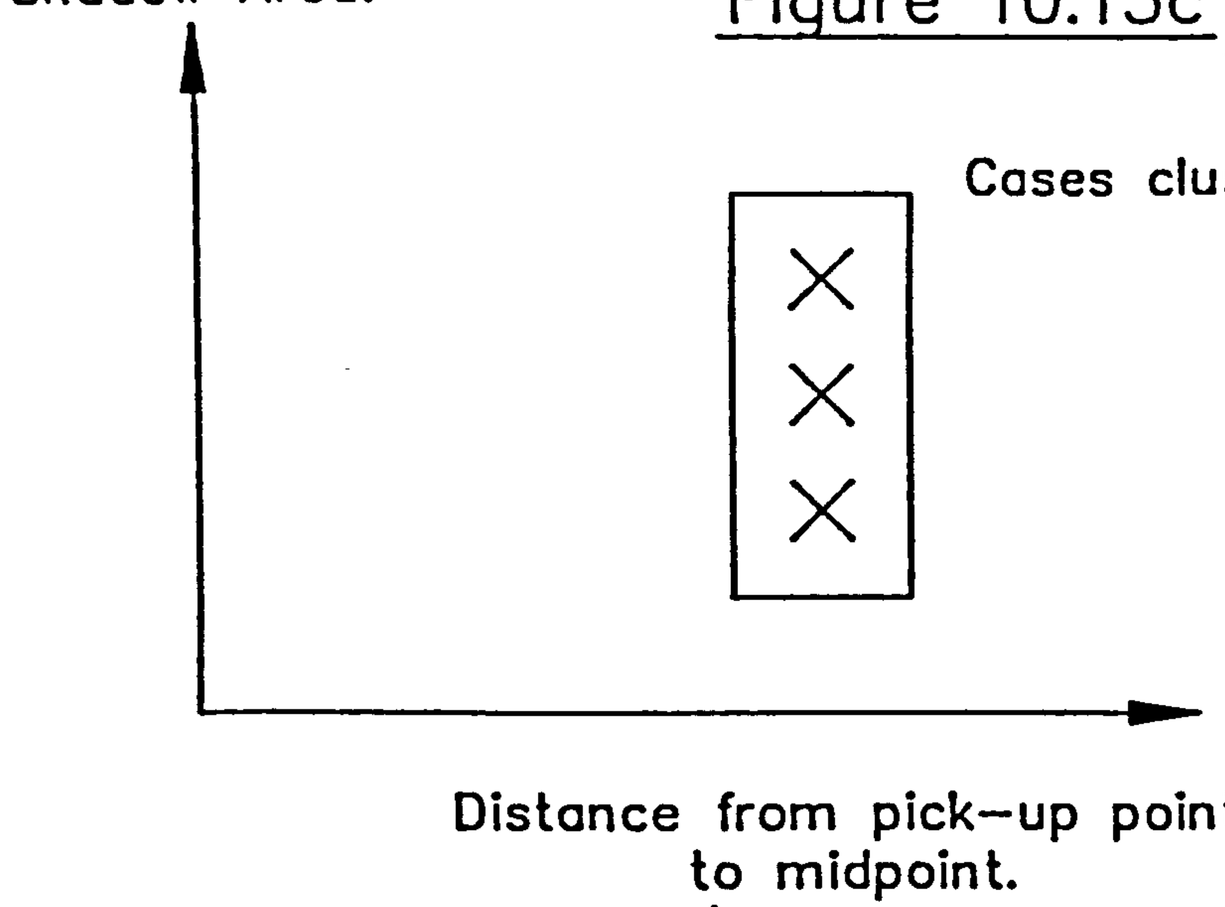


Figure 10.15c

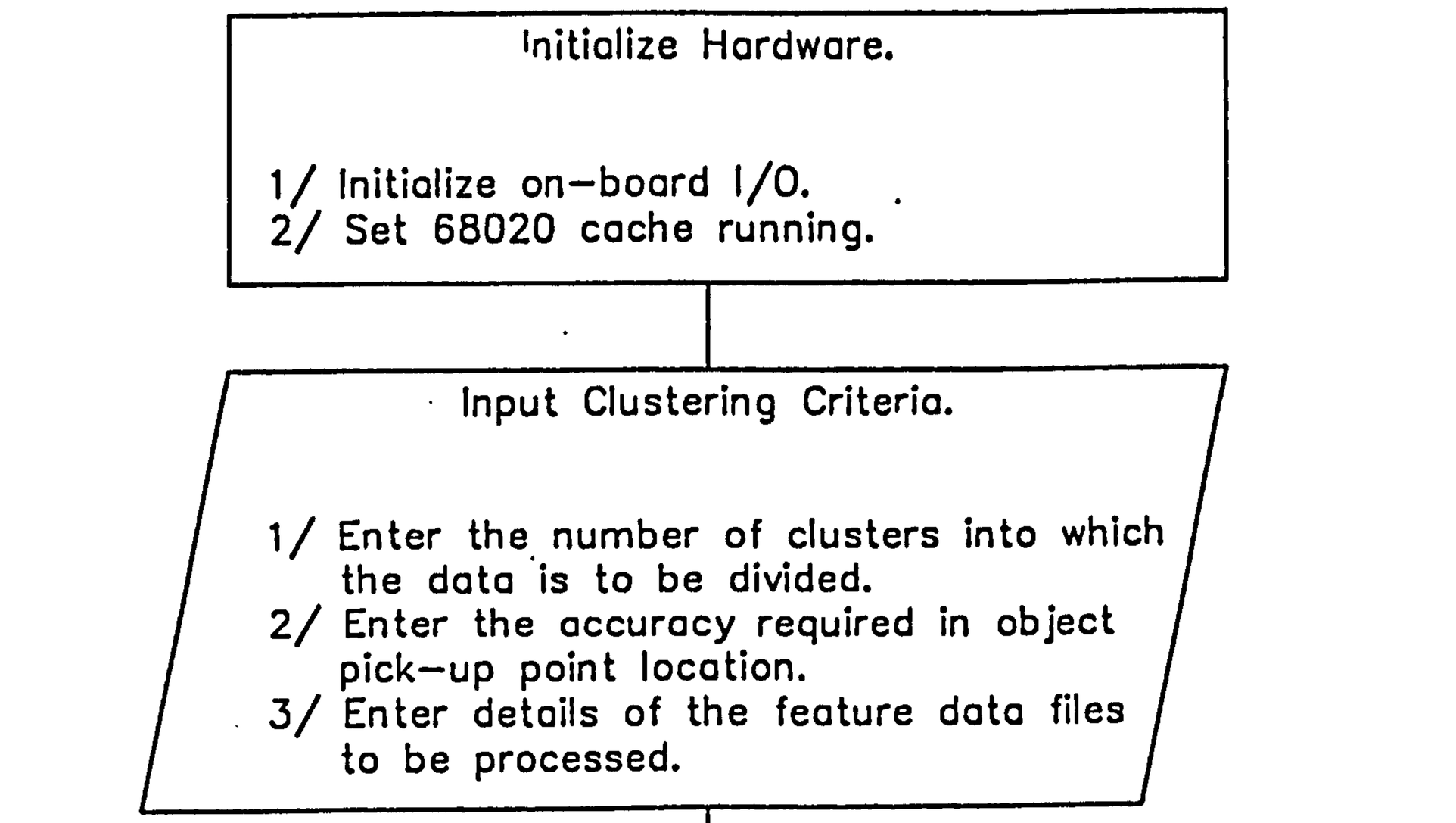


The effect of clustering the cases to provide only the required pick–up–point accuracy results in the production of a single cluster which contains feature vectors with large ranges in the identification features. This would be likely to cause ambiguity in object identification in the later stages of image processing.

CLUSTERING PROGRAM FOR DUMC 68020 UNIT.

Figure 10.16

Set-up phase.



Data processing phase.

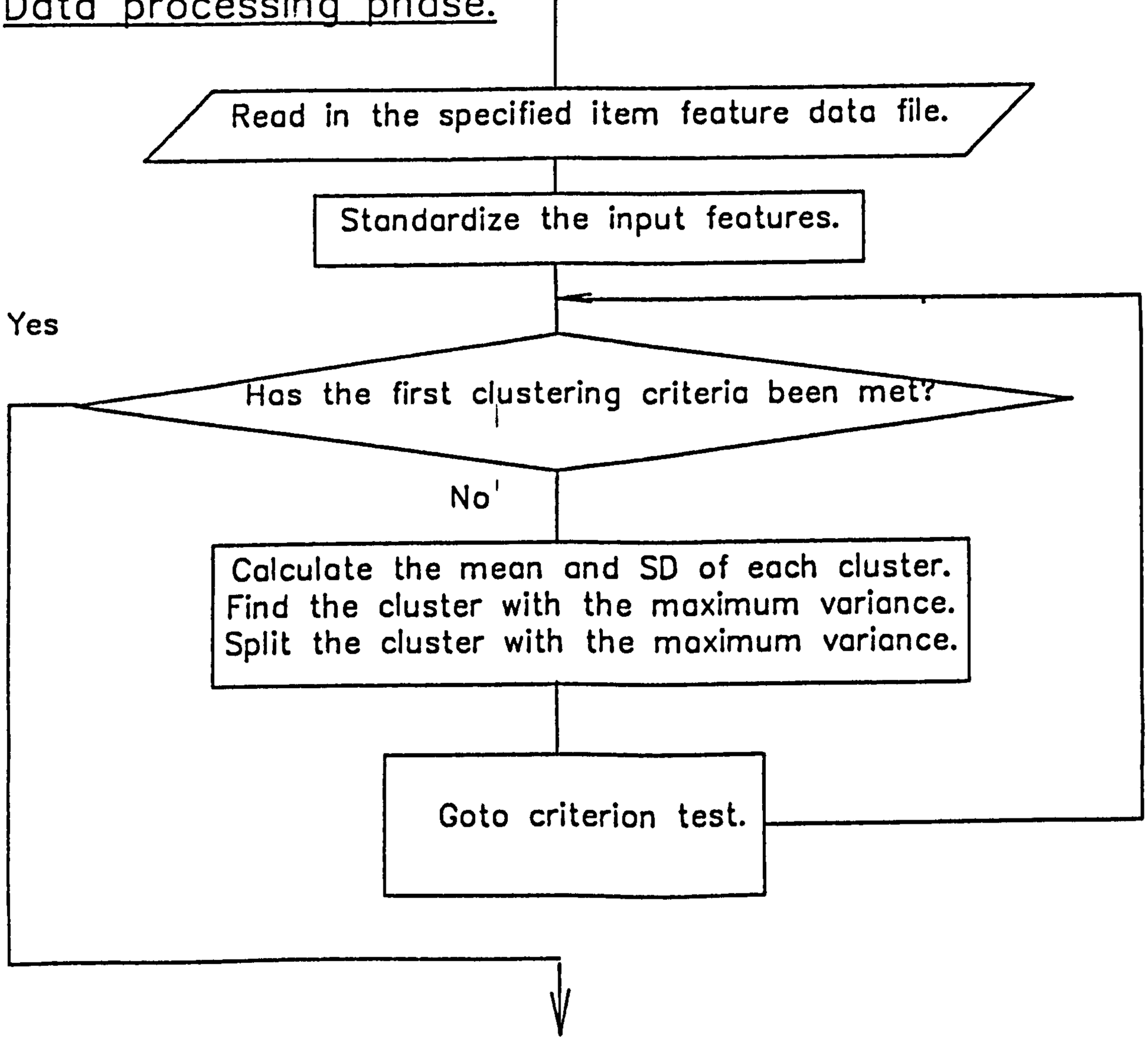
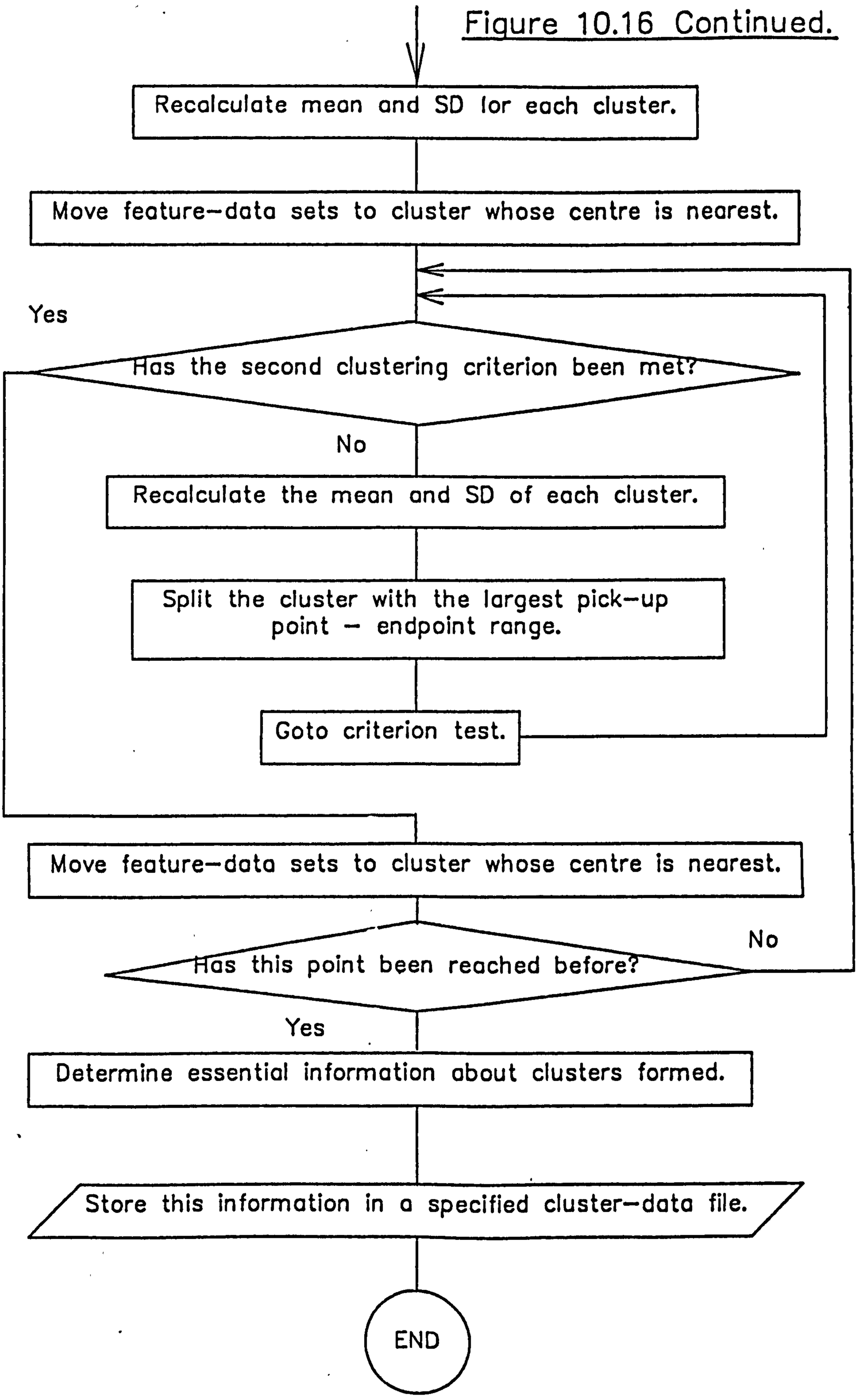


Figure 10.16 Continued.



```
**************************************************
*                                                *
*  ITEM CLUSTER DATA FILE GENERATION PROGRAM     *
*                                                *
**************************************************
```

Figure 10.17

```
The clustering sequence will be sent to the printer.
How many clusters do you want to divide your data into initially.
Enter the accuracy required in the robot pick-up point in mm.
The number of clusters the data is to be divided into initially is :-8.
The accuracy of pick up point required is 7 mm. which corresponds to 10 pixels.
Enter the name of the FEATURE data file to be read in.
Enter the set index of the final feature data to be read in.
Data sets with index up to 100 are to be read in from file "fnb1",
Drive 1 deselected.
All specified data read in from disc.

Averages of totals of variables are:- 86,86,51,56,255,757,49,106,23.
The value of sigma for variable 0 is 15.96
The value of sigma for variable 1 is 9.61
The value of sigma for variable 2 is 15.44
The value of sigma for variable 3 is 14.85
The value of sigma for variable 4 is 32.96
The value of sigma for variable 5 is 87.84
The value of sigma for variable 6 is 28.35
The value of sigma for variable 7 is 15.20
The value of sigma for variable 8 is 7.59
The value of sigma for variable 9 is 3.64
The value of sigma for variable 10 is 2.60
The value of sigma for variable 11 is 3.38
Standardization of data complete.

The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 1 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 1 .
Data sets assigned to new cluster are:
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
 41, 42, 59, 60, 61,
 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 97, 98, 99, 100,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 1, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 2 .
Data sets assigned to new cluster are:-
 19, 33, 35, 36, 42, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 97, 98, 99,

The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 3 .
Data sets assigned to new cluster are:-
 43, 45, 46, 55, 56, 57, 58,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 1, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 4 .
Data sets assigned to new cluster are:-
 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21,
 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 34, 37, 38, 39, 40, 41,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 3 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 5 .
Data sets assigned to new cluster are:-
 51, 52, 53, 54,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 2, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 2 has been split to form cluster 6 .
Data sets assigned to new cluster are:-
 19, 33, 35, 36, 42, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
```

```
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 2, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 2 has been split to form cluster 7 .
Data sets assigned to new cluster are:-
 97, 98,
The cluster with the feature with the largest variance has now been split.
The initial number of clusters has been reached.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 2, 3, 4, 5, 6, 24, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
43, 45, 46, 55, 56, 57, 58,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28,
29, 30, 31, 32, 34, 37, 38, 39, 40, 41,
Members of cluster 5 are:-
51, 52, 53, 54,
Members of cluster 6 are:-
19, 33, 35, 36, 42, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,

Members of cluster 7 are:-
97, 98,
About to enter movement stage.
Move routine entered.
Movement loop = 0.
Cluster change for data set 19 from cluster 6 to 4.
Movement loop = 1.
Cluster change for data set 24 from cluster 1 to 4.
Movement loop = 2.
Cluster change for data set 33 from cluster 6 to 4.
Movement loop = 3.
Cluster change for data set 35 from cluster 6 to 4.
Movement loop = 4.
Cluster change for data set 36 from cluster 6 to 4.
Movement loop = 5.
Cluster change for data set 42 from cluster 6 to 4.
Movement loop = 6.
Movement of points to nearest cluster centre completed.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 2, 3, 4, 5, 6, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
43, 45, 46, 55, 56, 57, 58,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
Members of cluster 5 are:-
51, 52, 53, 54,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
Members of cluster 7 are:-
97, 98,
Printout of initial cluster groupings complete.
About to enter range testing loop for the first time.
For cluster 0,feature 0, range is 1.
For cluster 0,feature 1, range is 1.
For cluster 0,feature 2, range is 4.
For cluster 1,feature 0, range is 4.
For cluster 1,feature 1, range is 11.
For cluster 1,feature 2, range is 1.
For cluster 2,feature 0, range is 0.
For cluster 2,feature 1, range is 0.
For cluster 2,feature 2, range is 0.
For cluster 3,feature 0, range is 12.
For cluster 3,feature 1, range is 2.
For cluster 3,feature 2, range is 14.
For cluster 4,feature 0, range is 3.
For cluster 4,feature 1, range is 4.
For cluster 4,feature 2, range is 12.
For cluster 5,feature 0, range is 9.
For cluster 5,feature 1, range is 3.
For cluster 5,feature 2, range is 1.
For cluster 6,feature 0, range is 12.
For cluster 6,feature 1, range is 6.
For cluster 6,feature 2, range is 1.
For cluster 7,feature 0, range is 1.
For cluster 7,feature 1, range is 5.
For cluster 7,feature 2, range is 2.
The maximum range (10) occurs for cluster 3, feature 2, range=14.
The maximum variance of cluster 3, is 0.2216 and occurs with feature 0.
Test finished to find cluster with maximum range in distance feature.
```

Figure 10.17 Continued.

Figure 10.17 Continued.

```
Cluster splitting routine entered.
Cluster 3 has been split to form cluster 8 .
Data sets assigned to new cluster are:-
 43, 57, 58,
The cluster with the largest range has been split.
The maximum range (10) occurs for cluster 8, feature 2, range=14.
The maximum variance of cluster 8, is 0.0709 and occurs with feature 2.
Test finished to find cluster with maximum range in distance feature.
Cluster splitting routine entered.
Cluster 8 has been split to form cluster 9 .
Data sets assigned to new cluster are:-
 57, 58,
The cluster with the largest range has been split.
The maximum range (10) occurs for cluster 3, feature 0, range=12.
The maximum variance of cluster 3, is 0.0666 and occurs with feature 3.
Test finished to find cluster with maximum range in distance feature.
Cluster splitting routine entered.
Cluster 3 has been split to form cluster 10 .
Data sets assigned to new cluster are:-
 55, 56,
The cluster with the largest range has been split.
The maximum range (10) occurs for cluster 4, feature 2, range=12.
The maximum variance of cluster 4, is 0.2128 and occurs with feature 0.
Test finished to find cluster with maximum range in distance feature.
Cluster splitting routine entered.
Cluster 4 has been split to form cluster 11 .
Data sets assigned to new cluster are:-
 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
The cluster with the largest range has been split.
The maximum range (10) occurs for cluster 6, feature 0, range=12.
The maximum variance of cluster 6, is 0.1517 and occurs with feature 1.
Test finished to find cluster with maximum range in distance feature.
Cluster splitting routine entered.
Cluster 6 has been split to form cluster 12 .
Data sets assigned to new cluster are:-
 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
The maximum range (10) occurs for cluster 1, feature 1, range=11.
The maximum variance of cluster 1, is 0.3413 and occurs with feature 3.
Test finished to find cluster with maximum range in distance feature.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 13 .
Data sets assigned to new cluster are:-
 2, 3, 4, 5, 6,
The cluster with the largest range has been split.
The maximum range (10) occurs for cluster 5, feature 0, range=9.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31,
Members of cluster 5 are:-
51, 52, 53, 54,
Members of cluster 6 are:-
59, 60, 61, 62,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
43,
Members of cluster 9 are:-
57, 58,
Members of cluster 10 are:-
55, 56,
Members of cluster 11 are:-
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
Members of cluster 12 are:-
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
Members of cluster 13 are:-
2, 3, 4, 5, 6,
Printout of distance cluster groupings complete.
Move routine entered.
Movement loop = 0.
Cluster change for data set 31 from cluster 4 to 11.
Movement loop = 1.
Cluster change for data set 54 from cluster 5 to 10.
Movement loop = 2.
Cluster change for data set 63 from cluster 12 to 6.
Movement loop = 3.
Cluster change for data set 64 from cluster 12 to 6.
Movement loop = 4.
Cluster change for data set 65 from cluster 12 to 6.
Movement loop = 5.
Cluster change for data set 66 from cluster 12 to 6.
Movement loop = 6.
Cluster change for data set 67 from cluster 12 to 6.
Movement loop = 7.
```

Figure 10.17 Continued.

```
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
43,
Members of cluster 9 are:-
57, 58,
Members of cluster 10 are:-
54, 55, 56,
Members of cluster 11 are:-
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
Members of cluster 12 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 13 are:-
2, 3, 4, 5, 6,
About to enter range testing loop for the second time.
For cluster 0,feature 0, range is 1.
For cluster 0,feature 1, range is 1.
For cluster 0,feature 2, range is 1.
For cluster 1,feature 0, range is 1.
For cluster 1,feature 1, range is 1.
For cluster 1,feature 2, range is 1.
For cluster 2,feature 0, range is 0.
For cluster 2,feature 1, range is 0.
For cluster 2,feature 2, range is 0.
For cluster 3,feature 0, range is 0.
For cluster 3,feature 1, range is 0.
For cluster 3,feature 2, range is 1.
For cluster 4,feature 0, range is 3.
For cluster 4,feature 1, range is 4.
For cluster 4,feature 2, range is 5.
For cluster 5,feature 0, range is 4.
For cluster 5,feature 1, range is 3.
For cluster 5,feature 2, range is 1.
For cluster 6,feature 0, range is 7.
For cluster 6,feature 1, range is 0.
For cluster 6,feature 2, range is 1.
For cluster 7,feature 0, range is 1.
For cluster 7,feature 1, range is 5.
For cluster 7,feature 2, range is 2.
For cluster 8,feature 0, range is 0.
For cluster 8,feature 1, range is 0.
For cluster 8,feature 2, range is 0.
For cluster 9,feature 0, range is 1.
For cluster 9,feature 1, range is 0.
For cluster 9,feature 2, range is 0.
For cluster 10,feature 0, range is 1.
For cluster 10,feature 1, range is 1.
For cluster 10,feature 2, range is 1.
For cluster 11,feature 0, range is 2.
For cluster 11,feature 1, range is 1.
For cluster 11,feature 2, range is 2.
For cluster 12,feature 0, range is 3.
For cluster 12,feature 1, range is 6.
For cluster 12,feature 2, range is 1.
For cluster 13,feature 0, range is 4.
For cluster 13,feature 1, range is 1.
For cluster 13,feature 2, range is 1.
The maximum range (10) occurs for cluster 6, feature 0, range=7.
```

Figure 10.17 Continued.

```
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
43,
Members of cluster 9 are:-
57, 58,
Members of cluster 10 are:-
54, 55, 56,
Members of cluster 11 are:-
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
Members of cluster 12 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 13 are:-
2, 3, 4, 5, 6.
Move routine entered.
Movement loop = 0.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
43,
Members of cluster 9 are:-
57, 58,
Members of cluster 10 are:-
54, 55, 56,
Members of cluster 11 are:-
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
Members of cluster 12 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 13 are:-
2, 3, 4, 5, 6.
Printout of final cluster groupings complete.
maxD= 46, minD= 45, cenD= 45.
maxD= 45, minD= 44, cenD= 44.
maxD= 45, minD= 41, cenD= 43.
For cluster 0 number of points is: 3.
Feature max and mins are:-
55, 47.  63, 61.  43, 29.  35, 21.  177, 163.  565, 553.  50, 47.  68, 64.  14, 8.
 The distances to the centre are:- 45,44,43.
maxD= 48, minD= 47, cenD= 47.
maxD= 34, minD= 33, cenD= 33.
maxD= 48, minD= 47, cenD= 47.
For cluster 1 number of points is: 1.
Feature max and mins are:-
59, 58.  87, 86.  35, 31.  58, 53.  218, 213.  802, 777.  100, 1.  89, 88.  9, 8.
 The distances to the centre are:- 47,33,47.
maxD= 49, minD= 49, cenD= 49.
maxD= 33, minD= 33, cenD= 33.
maxD= 48, minD= 48, cenD= 48.
```

Figure 10.17 Continued.

```
For cluster 2 number of points is: 0.
Feature max and mins are:-
61, 61.  88, 88.  52, 52.  39, 39.  217, 217.  770, 770.  99, 99.  91, 91.  9, 9.
 The distances to the centre are:- 49,33,48.
maxD= 46, minD= 46, cenD= 46.
maxD= 45, minD= 45, cenD= 45.
maxD= 34, minD= 33, cenD= 33.
For cluster 3 number of points is: 1.
Feature max and mins are:-
75, 71.  67, 67.  44, 39.  40, 39.  212, 206.  596, 592.  46, 45.  83, 79.  16, 15.
 The distances to the centre are:- 46,45,33.
maxD= 49, minD= 46, cenD= 47.
maxD= 47, minD= 43, cenD= 45.
maxD= 47, minD= 42, cenD= 44.
For cluster 4 number of points is: 23.
Feature max and mins are:-
95, 83.  94, 89.  46, 38.  76, 70.  284, 260.  866, 767.  30, 7.  119, 112.  34, 19.
 The distances to the centre are:- 47,45,44.
maxD= 44, minD= 40, cenD= 42.
maxD= 47, minD= 44, cenD= 45.
maxD= 46, minD= 45, cenD= 45.
For cluster 5 number of points is: 2.
Feature max and mins are:-
54, 47.  62, 61.  26, 18.  49, 39.  177, 163.  577, 554.  53, 51.  71, 65.  10, 9.
 The distances to the centre are:- 42,45,45.
maxD= 43, minD= 36, cenD= 39.
maxD= 46, minD= 46, cenD= 46.
maxD= 47, minD= 46, cenD= 46.
For cluster 6 number of points is: 8.
Feature max and mins are:-
104, 100.  89, 79.  68, 55.  49, 47.  278, 257.  749, 658.  67, 59.  116, 104.  22, 16.
 The distances to the centre are:- 39,46,46.
maxD= 49, minD= 48, cenD= 48.
maxD= 42, minD= 37, cenD= 39.
maxD= 48, minD= 46, cenD= 47.
For cluster 7 number of points is: 1.
Feature max and mins are:-
67, 62.  88, 88.  73, 64.  27, 21.  228, 221.  777, 767.  98, 97.  94, 91.  13, 10.
 The distances to the centre are:- 48,39,47.
maxD= 46, minD= 46, cenD= 46.
maxD= 45, minD= 45, cenD= 45.
maxD= 32, minD= 32, cenD= 32.
For cluster 8 number of points is: 0.
Feature max and mins are:-
87, 87.  72, 72.  41, 41.  50, 50.  235, 235.  633, 633.  43, 43.  91, 91.  17, 17.
 The distances to the centre are:- 46,45,32.
maxD= 35, minD= 34, cenD= 34.
maxD= 46, minD= 46, cenD= 46.
maxD= 46, minD= 46, cenD= 46.
For cluster 9 number of points is: 1.
Feature max and mins are:-
88, 87.  74, 73.  51, 47.  49, 44.  238, 236.  641, 635.  58, 57.  96, 95.  15, 14.
 The distances to the centre are:- 34,46,46.
maxD= 35, minD= 34, cenD= 34.
maxD= 47, minD= 46, cenD= 46.
maxD= 46, minD= 45, cenD= 45.
For cluster 10 number of points is: 2.
Feature max and mins are:-
74, 66.  70, 65.  39, 35.  48, 43.  214, 200.  615, 599.  56, 54.  87, 78.  13, 11.
 The distances to the centre are:- 34,46,45.
maxD= 48, minD= 46, cenD= 47.
maxD= 47, minD= 46, cenD= 46.
maxD= 42, minD= 35, cenD= 38.
For cluster 11 number of points is: 11.
Feature max and mins are:-
112, 96.  89, 79.  49, 43.  71, 59.  279, 260.  770, 658.  42, 31.  117, 106.  27, 19.
 The distances to the centre are:- 47,46,38.
maxD= 48, minD= 43, cenD= 45.
maxD= 47, minD= 41, cenD= 44.
maxD= 47, minD= 46, cenD= 46.
For cluster 12 number of points is: 25.
Feature max and mins are:-
100, 84.  94, 90.  74, 64.  49, 41.  284, 260.  837, 748.  93, 68.  117, 111.  33, 23.
 The distances to the centre are:- 45,44,46.
maxD= 49, minD= 45, cenD= 47.
maxD= 44, minD= 43, cenD= 43.
maxD= 48, minD= 47, cenD= 47.
For cluster 13 number of points is: 4.
Feature max and mins are:-
71, 58.  90, 87.  25, 17.  77, 72.  242, 216.  822, 798.  6, 2.  99, 92.  17, 11.
 The distances to the centre are:- 47,43,47.
Information to be placed in storage file has been determined.
Enter the name of the cluster data storage file.
```

```
**************************************************
*                                                *
*  ITEM CLUSTER DATA FILE GENERATION PROGRAM     *
*                                                *
**************************************************
```

Figure 10.18

```
The clustering sequence will be sent to the printer.
How many clusters do you want to divide your data into initially.
Enter the accuracy required in the robot pick-up point in mm.
The number of clusters the data is to be divided into initially is 1-14.
The accuracy of pick up point required is 20 mm. which corresponds to 30 pixels.
Enter the name of the FEATURE data file to be read in.
Enter the set index of the final feature data to be read in.
Data sets with index up to 100 are to be read in from file "fnb1".
Drive 1 deselected.
All specified data read in from disc.

Averages of totals of variables are:- 86,86,51,56,255,757,49,106,23.
The value of sigma for variable 0 is 15.96
The value of sigma for variable 1 is 9.61
The value of sigma for variable 2 is 15.44
The value of sigma for variable 3 is 14.85
The value of sigma for variable 4 is 32.96
The value of sigma for variable 5 is 87.84
The value of sigma for variable 6 is 28.35
The value of sigma for variable 7 is 15.20
The value of sigma for variable 8 is 7.59
The value of sigma for variable 9 is 3.64
The value of sigma for variable 10 is 2.60
The value of sigma for variable 11 is 1.38
Standardization of data complete.

The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 1 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 1 .
Data sets assigned to new cluster are:-
 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
 41, 42, 59, 60, 61,
 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 97, 98, 99, 100,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 1, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 2 .
Data sets assigned to new cluster are:-
 19, 33, 35, 36, 42, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 97, 98, 99,

The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 3 .
Data sets assigned to new cluster are:-
 43, 45, 46, 55, 56, 57, 58,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 1, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 4 .
Data sets assigned to new cluster are:-
 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21,
 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 34, 37, 38, 39, 40, 41,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 0, feature 3 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 0 has been split to form cluster 5 .
Data sets assigned to new cluster are:-
 51, 52, 53, 54,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 2, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 2 has been split to form cluster 6 .
Data sets assigned to new cluster are:-
 19, 33, 35, 36, 42, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
```

```
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 2, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 2 has been split to form cluster 7 .
Data sets assigned to new cluster are:
 97, 98,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 1, feature 3 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 1 has been split to form cluster 8 .
Data sets assigned to new cluster are:-
 2, 3, 4, 5, 6, 24,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 8, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 8 has been split to form cluster 9 .
Data sets assigned to new cluster are:-
 6, 24,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 9, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 9 has been split to form cluster 10 .
Data sets assigned to new cluster are:-
 24,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 6, feature 2 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 6 has been split to form cluster 11 .
Data sets assigned to new cluster are:-
 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,

The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 6, feature 3 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 6 has been split to form cluster 12 .
Data sets assigned to new cluster are:-
 19, 33, 35, 36,
The cluster with the feature with the largest variance has now been split.
The mean and variance of each cluster has now been calculated.
The maximum variance occurs with cluster 3, feature 0 .
Test finished to determine cluster containing the feature with the max variance.
Cluster splitting routine entered.
Cluster 3 has been split to form cluster 13 .
Data sets assigned to new cluster are:-
 43, 57, 58,
The cluster with the feature with the largest variance has now been split.
The initial number of clusters has been reached.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 55, 56,
Members of cluster 4 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28,
29, 30, 31, 32, 34, 37, 38, 39, 40, 41,
Members of cluster 5 are:-
51, 52, 53, 54,
Members of cluster 6 are:-
42, 59, 60, 61,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
24,
Members of cluster 11 are:-
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
19, 33, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
```

Figure 10.18 Continued.

```
About to enter movement stage.
Move routine entered.
Movement loop = 0.
Cluster change for data set 7 from cluster 4 to 10
Movement loop = 1.
Cluster change for data set 8 from cluster 4 to 10.
Movement loop = 2.
Cluster change for data set 18 from cluster 4 to 10.
Movement loop = 3.
Cluster change for data set 14 from cluster 4 to 10.
Movement loop = 4.
Cluster change for data set 9 from cluster 4 to 10.
Movement loop = 5.
Cluster change for data set 10 from cluster 4 to 10.
Movement loop = 6.
Cluster change for data set 11 from cluster 4 to 10.
Movement loop = 7.
Cluster change for data set 12 from cluster 4 to 10.
Movement loop = 8.
Cluster change for data set 13 from cluster 4 to 10.
Movement loop = 9.
Cluster change for data set 15 from cluster 4 to 10.
Movement loop = 10.
Cluster change for data set 16 from cluster 4 to 10.
Movement loop = 11.
Cluster change for data set 17 from cluster 4 to 10.
Movement loop = 12.
Cluster change for data set 19 from cluster 12 to 10.
Movement loop = 13.
Cluster change for data set 20 from cluster 4 to 10.
Movement loop = 14.
Cluster change for data set 21 from cluster 4 to 10.
Movement loop = 15.
Cluster change for data set 22 from cluster 4 to 10.
Movement loop = 16.
Cluster change for data set 23 from cluster 4 to 10.
Movement loop = 17.
Cluster change for data set 25 from cluster 4 to 10.
Movement loop = 18.
Cluster change for data set 26 from cluster 4 to 10.
Movement loop = 19.
Cluster change for data set 27 from cluster 4 to 10.
Movement loop = 20.
Cluster change for data set 28 from cluster 4 to 10.
Movement loop = 21.
Cluster change for data set 29 from cluster 4 to 10.
Movement loop = 22.
Cluster change for data set 30 from cluster 4 to 12.
Movement loop = 23.
Cluster change for data set 31 from cluster 4 to 12.
Movement loop = 24.
Cluster change for data set 32 from cluster 4 to 12.
Movement loop = 25.
Cluster change for data set 34 from cluster 4 to 12.
Movement loop = 26.
Cluster change for data set 54 from cluster 5 to 3.
Movement loop = 27.
Cluster change for data set 62 from cluster 11 to 6.
Movement loop = 28.
Cluster change for data set 42 from cluster 6 to 4.
Movement loop = 29.
Cluster change for data set 63 from cluster 11 to 6.
Movement of points to nearest cluster centre completed.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 54, 55, 56,
Members of cluster 4 are:-
37, 38, 39, 40, 41, 42,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29,
Members of cluster 11 are:-
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
30, 31, 32, 33, 34, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
Printout of initial cluster groupings complete.
```

Figure 10.18 Continued.

```
About to enter range testing loop for the first time.
For cluster 0,feature 0, range is 1.
For cluster 0,feature 1, range is 1.
For cluster 0,feature 2, range is 4.
For cluster 1,feature 0, range is 1.
For cluster 1,feature 1, range is 1.
For cluster 1,feature 2, range is 1.
For cluster 2,feature 0, range is 0.
For cluster 2,feature 1, range is 0.
For cluster 2,feature 2, range is 0.
For cluster 3,feature 0, range is 12.
For cluster 3,feature 1, range is 2.
For cluster 3,feature 2, range is 13.
For cluster 4,feature 0, range is 1.
For cluster 4,feature 1, range is 0.
For cluster 4,feature 2, range is 4.
For cluster 5,feature 0, range is 4.
For cluster 5,feature 1, range is 3.
For cluster 5,feature 2, range is 1.
For cluster 6,feature 0, range is 6.
For cluster 6,feature 1, range is 0.
For cluster 6,feature 2, range is 1.
For cluster 7,feature 0, range is 1.
For cluster 7,feature 1, range is 5.
For cluster 7,feature 2, range is 2.
For cluster 8,feature 0, range is 2.
For cluster 8,feature 1, range is 1.
For cluster 8,feature 2, range is 1.
For cluster 9,feature 0, range is 0.
For cluster 9,feature 1, range is 0.
For cluster 9,feature 2, range is 0.
For cluster 10,feature 0, range is 3.
For cluster 10,feature 1, range is 4.
For cluster 10,feature 2, range is 4.
For cluster 11,feature 0, range is 6.
For cluster 11,feature 1, range is 6.
For cluster 11,feature 2, range is 1.
For cluster 12,feature 0, range is 1.
For cluster 12,feature 1, range is 1.
For cluster 12,feature 2, range is 2.
For cluster 13,feature 0, range is 12.
For cluster 13,feature 1, range is 1.
For cluster 13,feature 2, range is 14.
The maximum range (30) occurs for cluster 13, feature 2, range=14.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 54, 55, 56,
Members of cluster 4 are:-
37, 38, 39, 40, 41, 42,
Members of cluster 5 are:
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63,
Members of cluster 7 are:
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29,
Members of cluster 11 are:-
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
30, 31, 32, 33, 34, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
Printout of distance cluster groupings complete.
Move routine entered.
Movement loop = 0.
Cluster change for data set 64 from cluster 11 to 6.
Movement loop = 1.
Cluster change for data set 65 from cluster 11 to 6.
Movement loop = 2.
Cluster change for data set 66 from cluster 11 to 6.
Movement loop = 3.
Cluster change for data set 67 from cluster 11 to 6.
Movement loop = 4.
```

Figure 10.18 Continued.

```
Members of cluster 0 are:-
47, 48, 49,. 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 54, 55, 56,
Members of cluster 4 are:-
37, 38, 39, 40, 41, 42,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29,
Members of cluster 11 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
30, 31, 32, 33, 34, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
About to enter range testing loop for the second time.
For cluster 0,feature 0, range is 1.
For cluster 0,feature 1, range is 1.
For cluster 0,feature 2, range is 4.
For cluster 1,feature 0, range is 1.
For cluster 1,feature 1, range is 1.
For cluster 1,feature 2, range is 1.
For cluster 2,feature 0, range is 0.
For cluster 2,feature 1, range is 0.
For cluster 2,feature 2, range is 0.
For cluster 3,feature 0, range is 12.
For cluster 3,feature 1, range is 2.
For cluster 3,feature 2, range is 13.
For cluster 4,feature 0, range is 1.
For cluster 4,feature 1, range is 0.
For cluster 4,feature 2, range is 4.
For cluster 5,feature 0, range is 4.
For cluster 5,feature 1, range is 3.
For cluster 5,feature 2, range is 1.
For cluster 6,feature 0, range is 7.
For cluster 6,feature 1, range is 0.
For cluster 6,feature 2, range is 1.
For cluster 7,feature 0, range is 1.
For cluster 7,feature 1, range is 5.
For cluster 7,feature 2, range is 2.
For cluster 8,feature 0, range is 2.
For cluster 8,feature 1, range is 1.
For cluster 8,feature 2, range is 1.
For cluster 9,feature 0, range is 0.
For cluster 9,feature 1, range is 0.
For cluster 9,feature 2, range is 0.
For cluster 10,feature 0, range is 3.
For cluster 10,feature 1, range is 4.
For cluster 10,feature 2, range is 4.
For cluster 11,feature 0, range is 5.
For cluster 11,feature 1, range is 6.
For cluster 11,feature 2, range is 1.
For cluster 12,feature 0, range is 1.
For cluster 12,feature 1, range is 1.
For cluster 12,feature 2, range is 2.
For cluster 13,feature 0, range is 12.
For cluster 13,feature 1, range is 1.
For cluster 13,feature 2, range is 14.
The maximum range (30) occurs for cluster 13, feature 2, range=14.
```

Figure 10.18 Continued.

Figure 10.18 Continued.

```
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 54, 55, 56,
Members of cluster 4 are:-
37, 38, 39, 40, 41, 42,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29,
Members of cluster 11 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
30, 31, 32, 33, 34, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
Move routine entered.
Movement loop = 0.
Members of cluster 0 are:-
47, 48, 49, 50,
Members of cluster 1 are:-
1, 100,
Members of cluster 2 are:-
99,
Members of cluster 3 are:-
45, 46, 54, 55, 56,
Members of cluster 4 are:-
37, 38, 39, 40, 41, 42,
Members of cluster 5 are:-
51, 52, 53,
Members of cluster 6 are:-
59, 60, 61, 62, 63, 64, 65, 66, 67,
Members of cluster 7 are:-
97, 98,
Members of cluster 8 are:-
2, 3, 4, 5,
Members of cluster 9 are:-
6,
Members of cluster 10 are:-
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29,
Members of cluster 11 are:-
68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93,
Members of cluster 12 are:-
30, 31, 32, 33, 34, 35, 36,
Members of cluster 13 are:-
43, 57, 58,
Printout of final cluster groupings complete.
maxD= 46, minD= 45, cenD= 45.
maxD= 45, minD= 44, cenD= 44.
maxD= 45, minD= 41, cenD= 43.
For cluster 0 number of points is: 3.
Feature max and mins are:-
55, 47.  63, 61.  43, 29.  35, 21.  177, 163.  565, 553.  50, 47.  68, 64.  14, 8.
 The distances to the centre are:- 45,44,43.
maxD= 48, minD= 47, cenD= 47.
maxD= 34, minD= 33, cenD= 33.
maxD= 48, minD= 47, cenD= 47.
For cluster 1 number of points is: 1.
Feature max and mins are:-
59, 58.  87, 86.  35, 31.  58, 53.  218, 213.  802, 777.  100, 1.  89, 88.  9, 8.
 The distances to the centre are:- 47,33,47.
maxD= 49, minD= 49, cenD= 49.
maxD= 33, minD= 33, cenD= 33.
maxD= 48, minD= 48, cenD= 48.
```

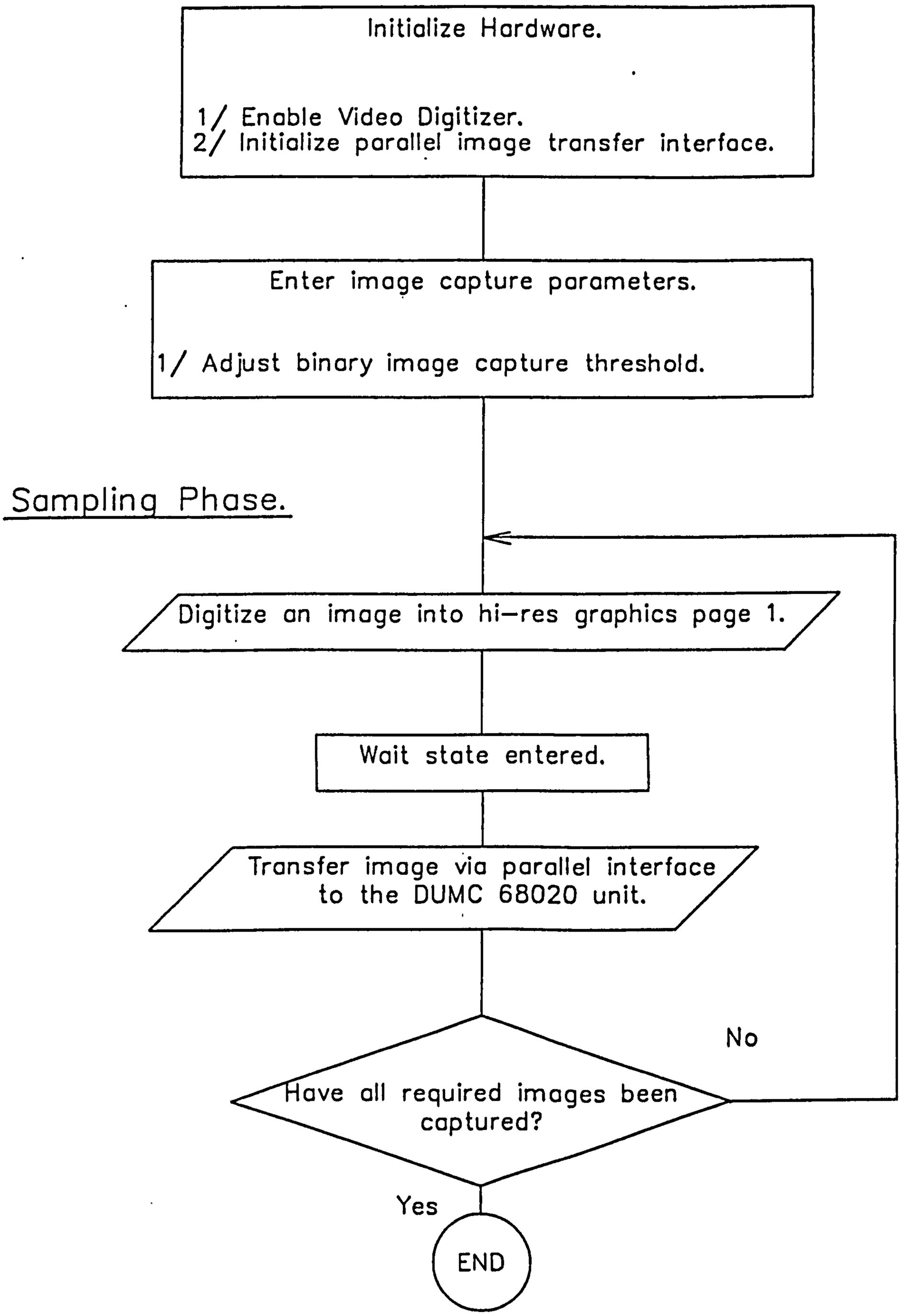
Figure 10.18 Continued.

```
For cluster 2 number of points is: 0.
Feature max and mins are:-
61, 61.  88, 88.  52, 52.  39, 39.  217, 217.  770, 770.  99, 99.  91, 91.  9, 9.
 The distances to the centre are:- 49,33,48.
maxD= 46, minD= 34, cenD= 40.
maxD= 47, minD= 45, cenD= 46.
maxD= 46, minD= 33, cenD= 39.
For cluster 3 number of points is: 4.
Feature max and mins are:-
75, 66.  70, 65.  44, 35.  48, 39.  214, 200.  615, 592.  56, 45.  87, 78.  16, 11.
 The distances to the centre are:- 40,46,39.
maxD= 47, minD= 46, cenD= 46.
maxD= 46, minD= 46, cenD= 46.
maxD= 39, minD= 35, cenD= 37.
For cluster 4 number of points is: 5.
Feature max and mins are:-
112, 101.  86, 79.  47, 43.  69, 59.  279, 260.  722, 658.  42, 37.  113, 106.  24, 19.
 The distances to the centre are:- 46,46,37.
maxD= 44, minD= 40, cenD= 42.
maxD= 47, minD= 44, cenD= 45.
maxD= 46, minD= 45, cenD= 45.
For cluster 5 number of points is: 2.
Feature max and mins are:-
54, 47.  62, 61.  26, 18.  49, 39.  177, 163.  577, 554.  53, 51.  71, 65.  10, 9.
 The distances to the centre are:- 42,45,45.
maxD= 43, minD= 36, cenD= 39.
maxD= 46, minD= 46, cenD= 46.
maxD= 47, minD= 46, cenD= 46.
For cluster 6 number of points is: 8.
Feature max and mins are:-
104, 100.  89, 79.  68, 55.  49, 47.  278, 257.  749, 658.  67, 59.  116, 104.  22, 16.
 The distances to the centre are:- 39,46,46.
maxD= 49, minD= 48, cenD= 48.
maxD= 42, minD= 37, cenD= 39.
maxD= 48, minD= 46, cenD= 47.
For cluster 7 number of points is: 1.
Feature max and mins are:-
67, 62.  88, 88.  73, 64.  27, 21.  228, 221.  777, 767.  98, 97.  94, 91.  13, 10.
 The distances to the centre are:- 48,39,47.
maxD= 49, minD= 47, cenD= 48.
maxD= 44, minD= 43, cenD= 43.
maxD= 48, minD= 47, cenD= 47.
For cluster 8 number of points is: 3.
Feature max and mins are:-
64, 58.  90, 87.  21, 17.  77, 72.  229, 216.  842, 798.  5, 2.  97, 92.  15, 11.
 The distances to the centre are:- 48,43,47.
maxD= 45, minD= 45, cenD= 45.
maxD= 44, minD= 44, cenD= 44.
maxD= 47, minD= 47, cenD= 47.
For cluster 9 number of points is: 0.
Feature max and mins are:-
71, 71.  88, 88.  25, 25.  74, 74.  242, 242.  817, 817.  6, 6.  99, 99.  17, 17.
 The distances to the centre are:- 45,44,47.
maxD= 49, minD= 46, cenD= 47.
maxD= 47, minD= 43, cenD= 45.
maxD= 47, minD= 43, cenD= 45.
For cluster 10 number of points is: 22.
Feature max and mins are:-
93, 83.  94, 90.  46, 38.  76, 70.  284, 260.  866, 779.  29, 7.  119, 112.  34, 19.
 The distances to the centre are:- 47,45,45.
maxD= 48, minD= 43, cenD= 45.
maxD= 47, minD= 41, cenD= 44.
maxD= 47, minD= 46, cenD= 46.
For cluster 11 number of points is: 25.
Feature max and mins are:-
100, 84.  94, 90.  74, 64.  49, 41.  284, 260.  837, 748.  93, 68.  117, 111.  33, 23.
 The distances to the centre are:- 45,44,46.
maxD= 48, minD= 47, cenD= 47.
maxD= 47, minD= 46, cenD= 46.
maxD= 42, minD= 40, cenD= 41.
For cluster 12 number of points is: 6.
Feature max and mins are:-
104, 95.  89, 87.  49, 44.  71, 68.  279, 270.  770, 746.  36, 30.  117, 114.  28, 25.
 The distances to the centre are:- 47,46,41.
maxD= 46, minD= 34, cenD= 40.
maxD= 46, minD= 45, cenD= 45.
maxD= 46, minD= 32, cenD= 39.
For cluster 13 number of points is: 2.
Feature max and mins are:-
88, 87.  74, 72.  51, 41.  50, 44.  238, 235.  641, 633.  58, 43.  96, 91.  17, 14.
 The distances to the centre are:- 40,45,39.
Information to be placed in storage file has been determined.
Enter the name of the cluster data storage file.
```

APPLE PROGRAM FOR TEST IMAGE CAPTURE.

Figure 10.19

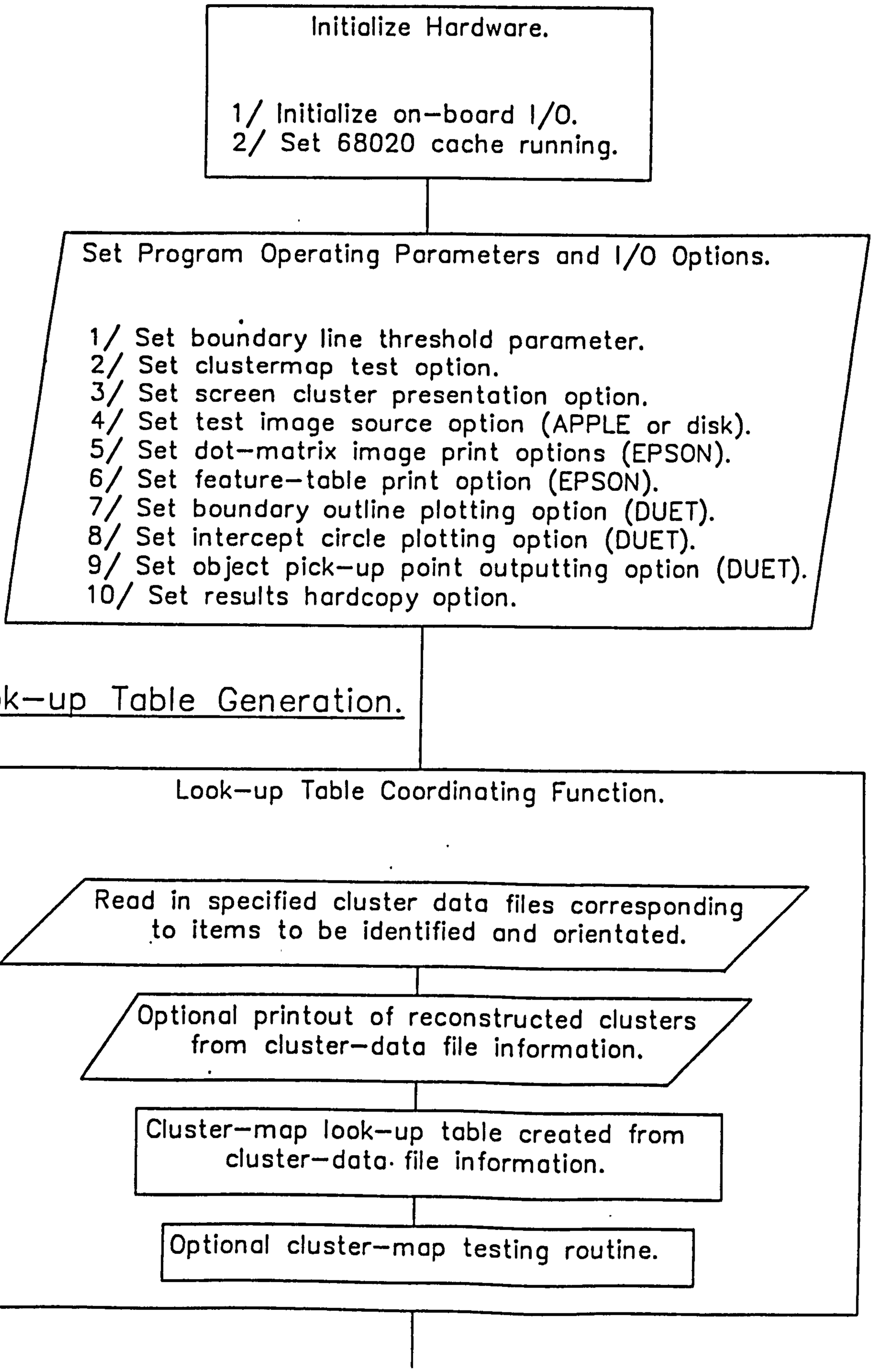
Set-up Phase.



COMPLETE IMAGE TESTING SEQUENCE WITH OBJECT IDENTIFICATION AND ORIENTATION MADE WITH REFERENCE TO CLUSTER-MAPPED LOOK-UP TABLE.

Set-up Phase.

Figure 10.20



Complete Image Testing Sequence Continued.

Image Testing Phase.

Figure 10.20 Continued.

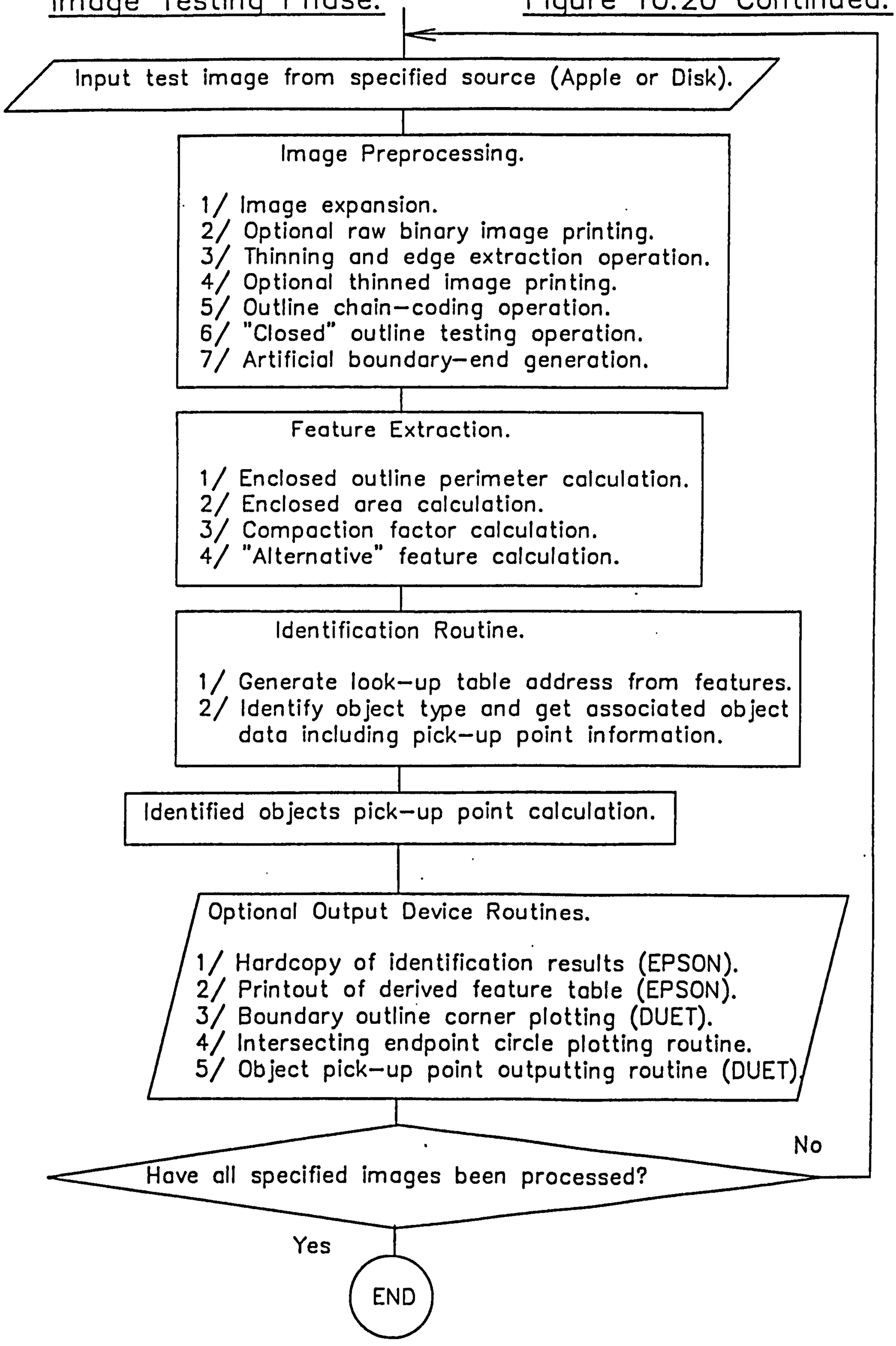


Diagram Illustrating the Calculation of Object Pick-up Points.

Figure 11.1

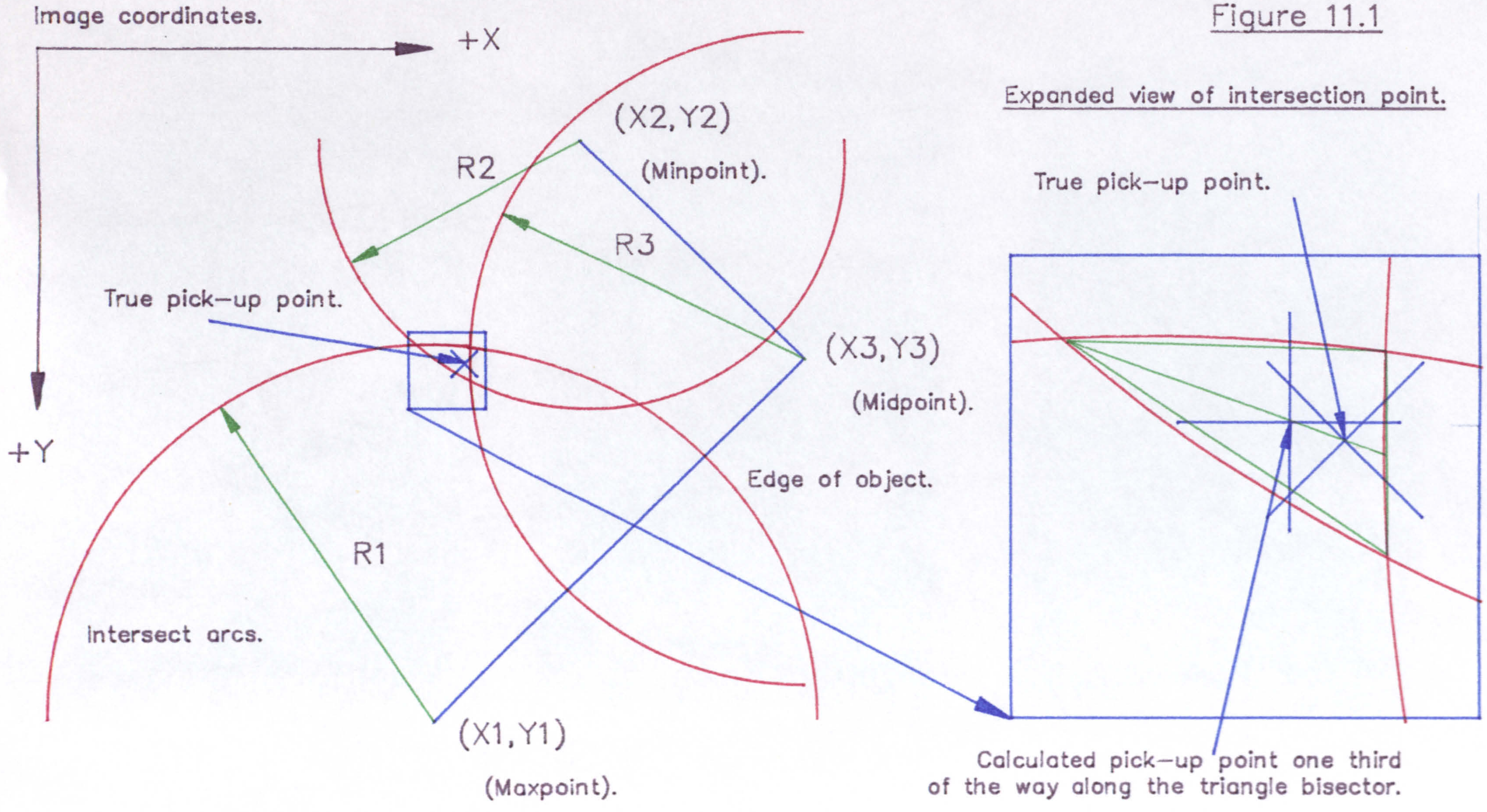


Figure 12.1

Figure 12.2

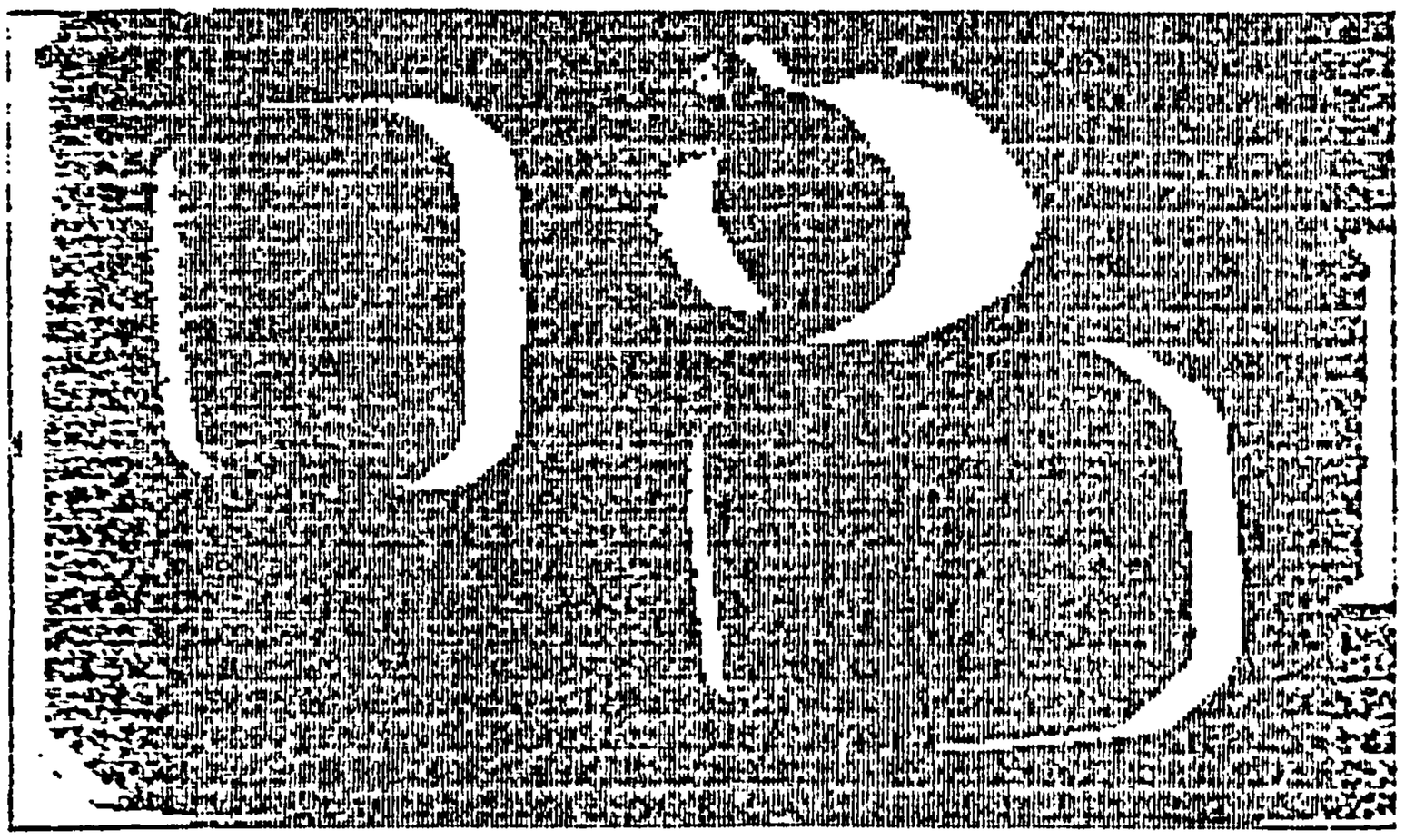
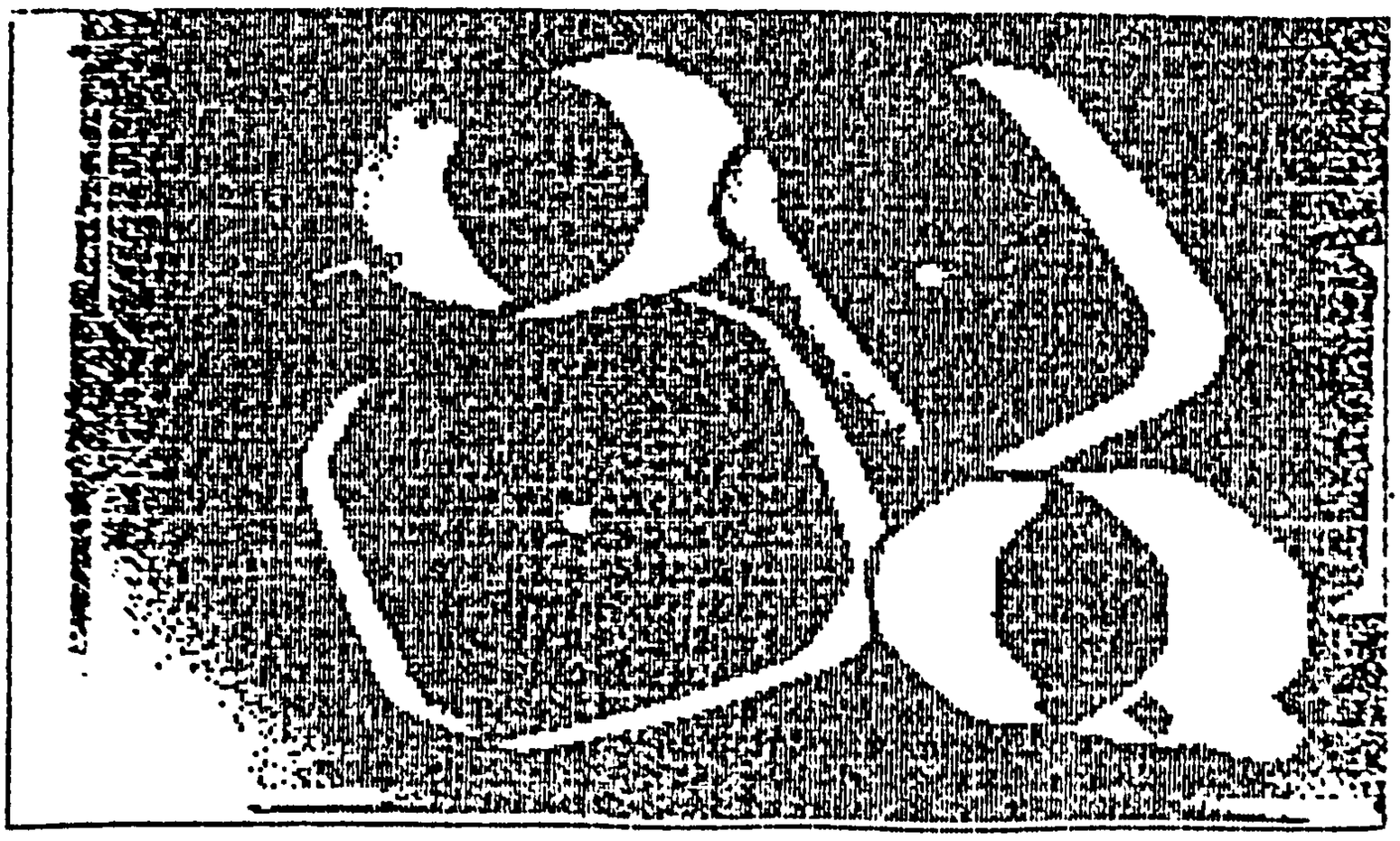


Figure 12.3



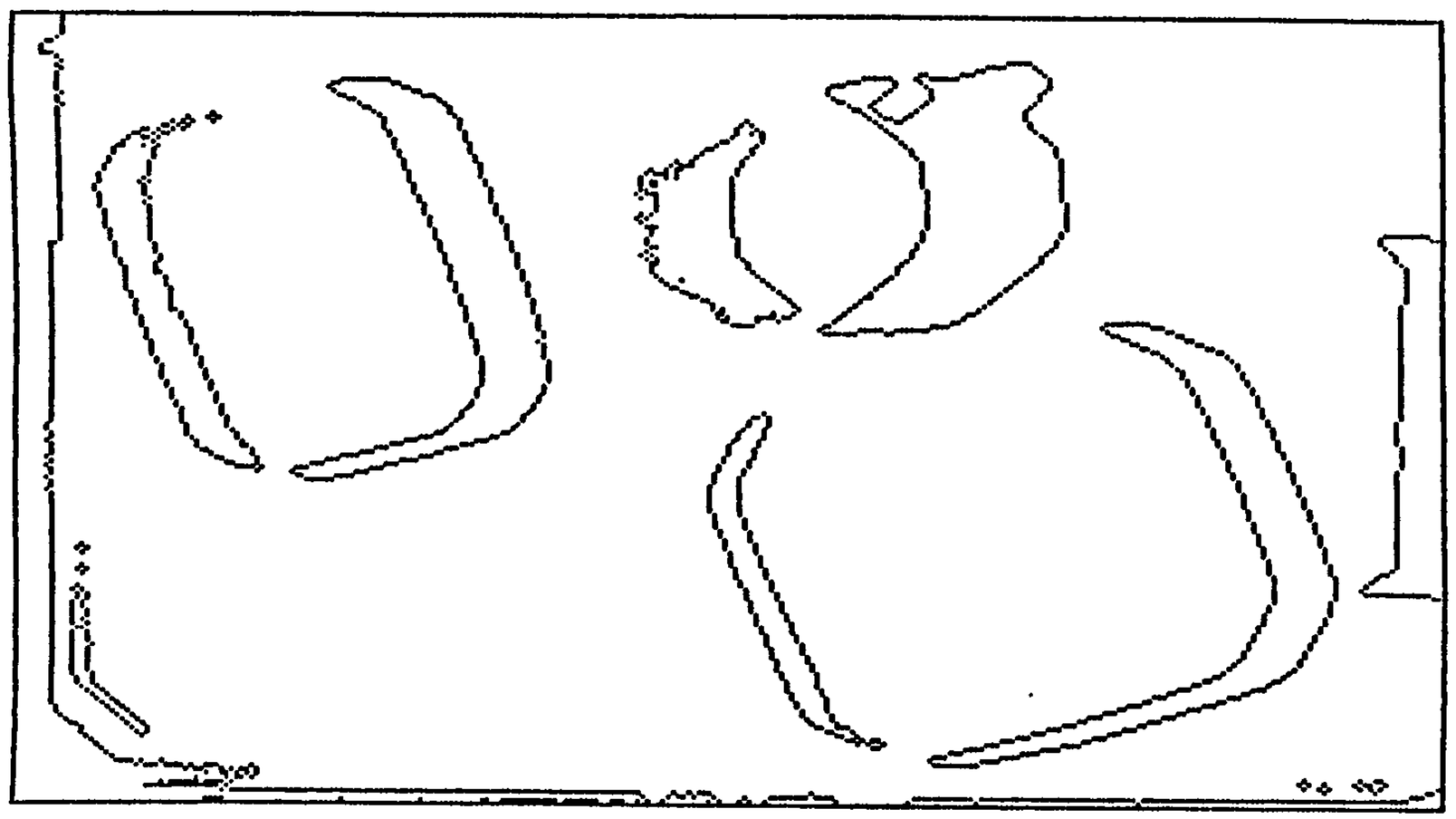


Figure 12.1a

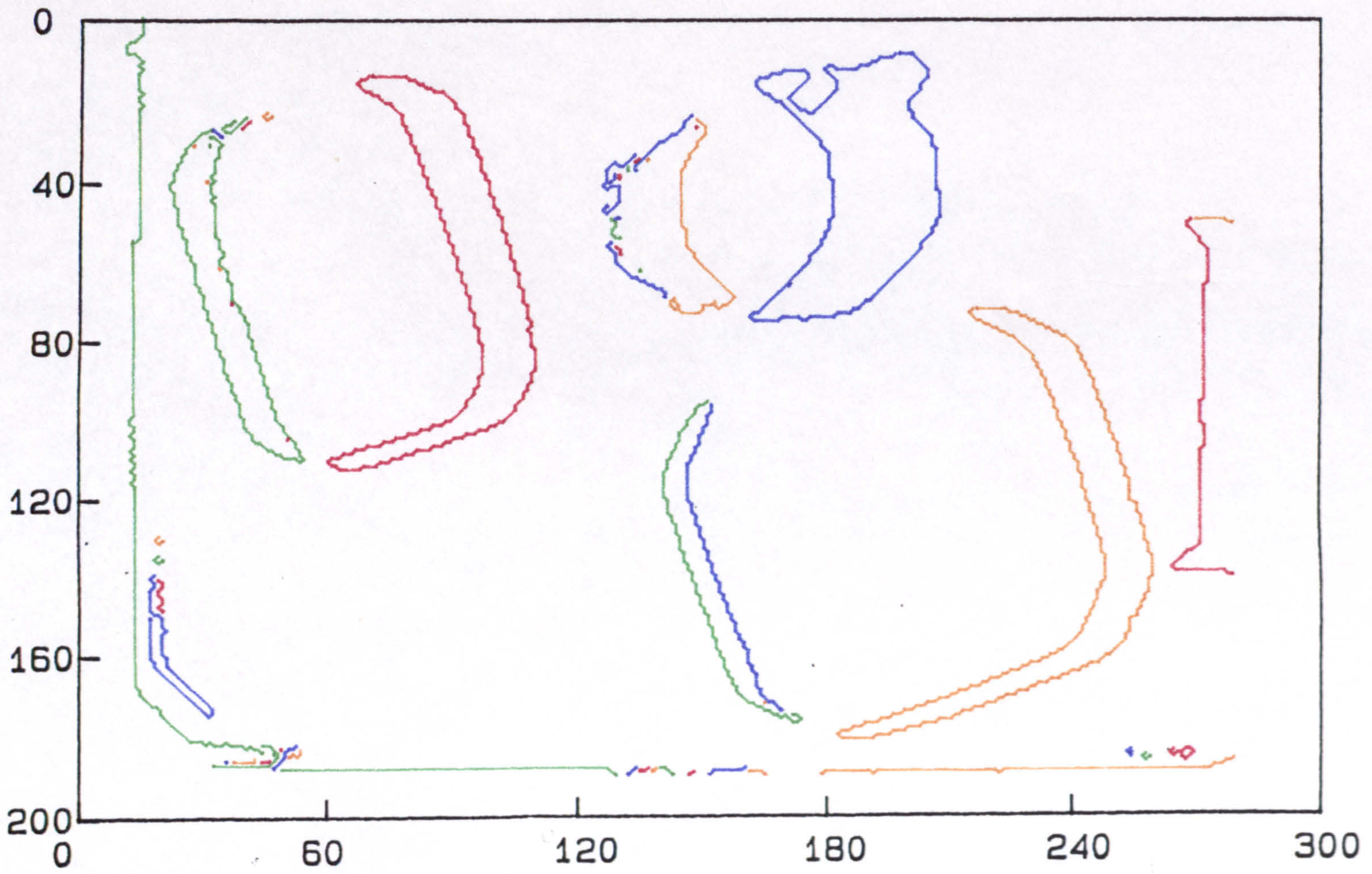
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	205	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	212	256	1706	38	0	0	0	0	0	0
2	62	17	3	92	83	72	57	111	118	265	304	1403	65	-5	94	72	44	116	94
8	15	43	16	15	43	16	15	43	16	175	203	730	56	0	0	0	0	0	0
22	268	54	2	273	60	8	265	138	87	104	0	0	0	-3	84	7	78	85	84
31	213	75	1	247	136	67	182	180	145	320	370	1422	96	-31	105	69	78	147	109
32	0	0	0	0	0	0	0	0	0	93	0	0	0	0	0	0	0	0	0
51	219	190	60	279	187	0	0	0	0	105	0	0	0	-219	-190	60	335	395	289
52	122	189	80	122	189	80	122	189	80	83	0	0	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.1a1



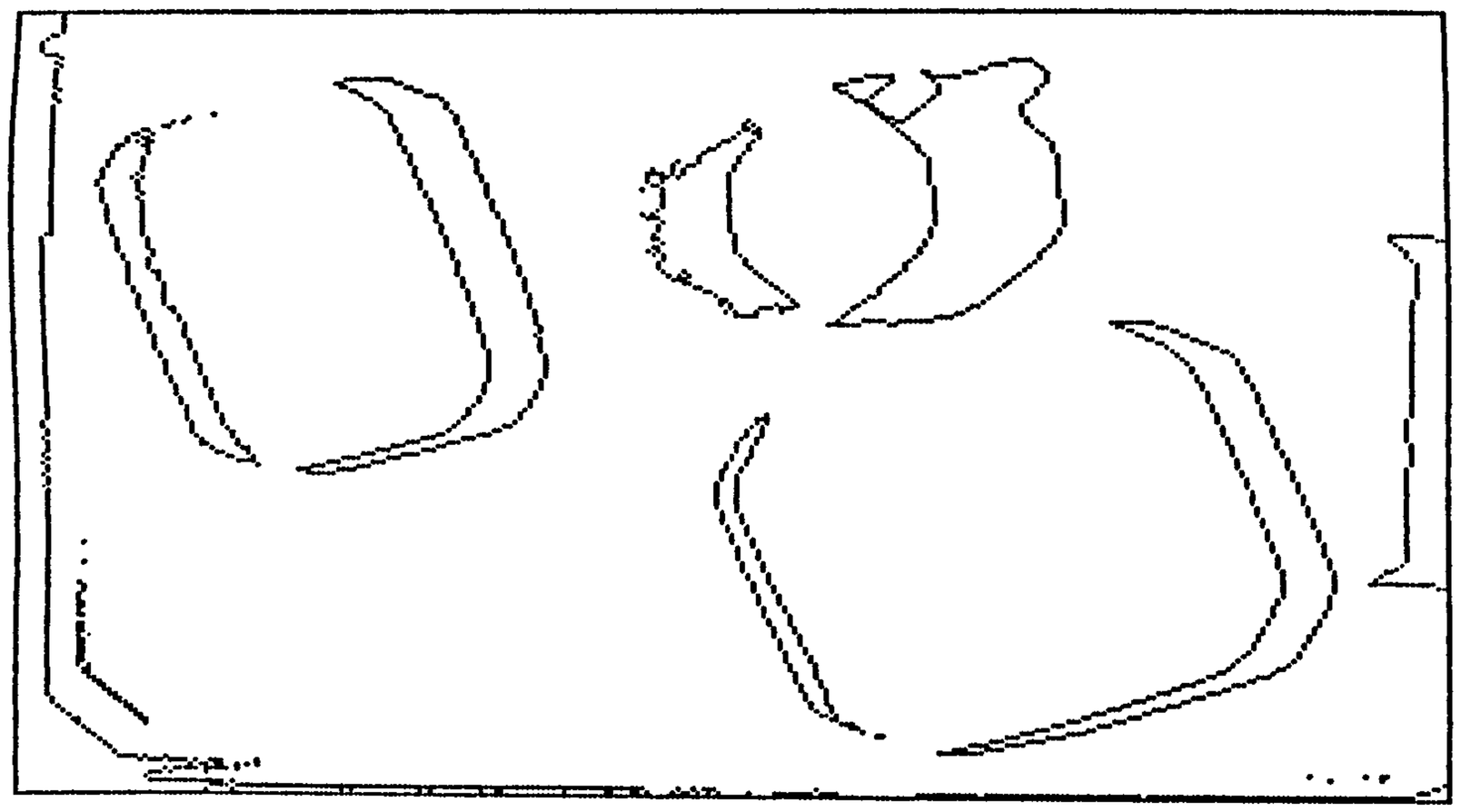


Figure 12.1b

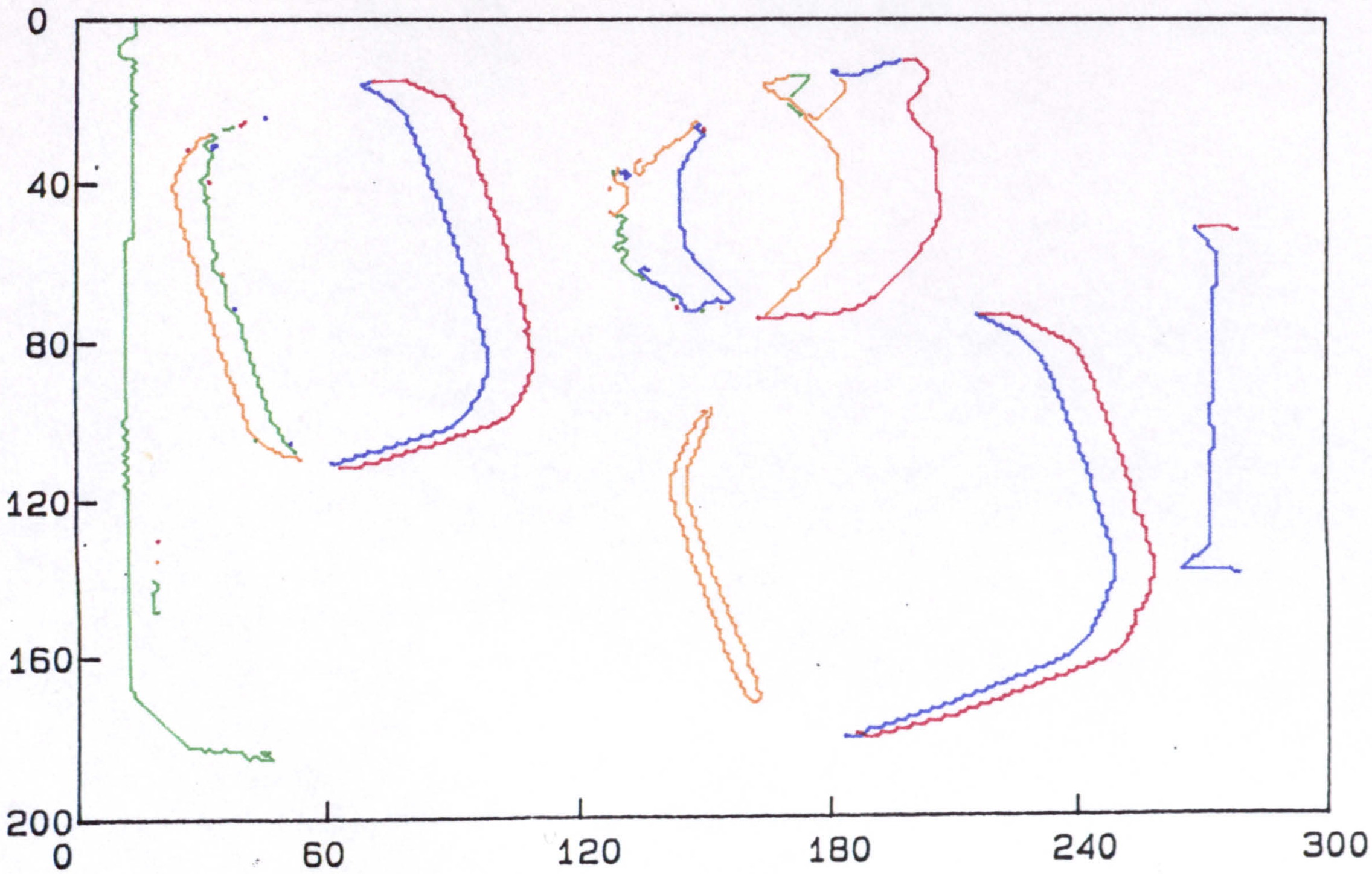
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	214	0	0	0	0	0	0	0	0	0
2	199	11	3	205	45	37	172	76	79	93	0	0	0	-27	65	34	45	79	70
5	63	17	2	93	83	71	57	112	118	121	0	0	0	-6	95	72	46	118	95
6	74	16	8	104	84	81	62	113	133	139	0	0	0	-12	97	74	51	125	97
15	0	0	0	0	0	0	0	0	0	89	0	0	0	0	0	0	0	0	0
16	26	29	0	27	30	1	27	60	32	82	0	0	0	1	31	1	30	31	31
25	269	54	1	274	60	7	266	138	86	102	0	0	0	-3	84	7	78	85	84
33	214	75	0	248	136	66	184	181	143	148	0	0	0	-30	106	69	78	147	110
34	222	75	7	258	136	76	188	181	160	166	0	0	0	-34	106	70	83	153	111
35	137	121	126	141	105	126	137	115	116	150	113	43	296	0	-6	16	10	26	6
52	24	188	4	24	188	4	24	188	4	104	0	0	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.1b1



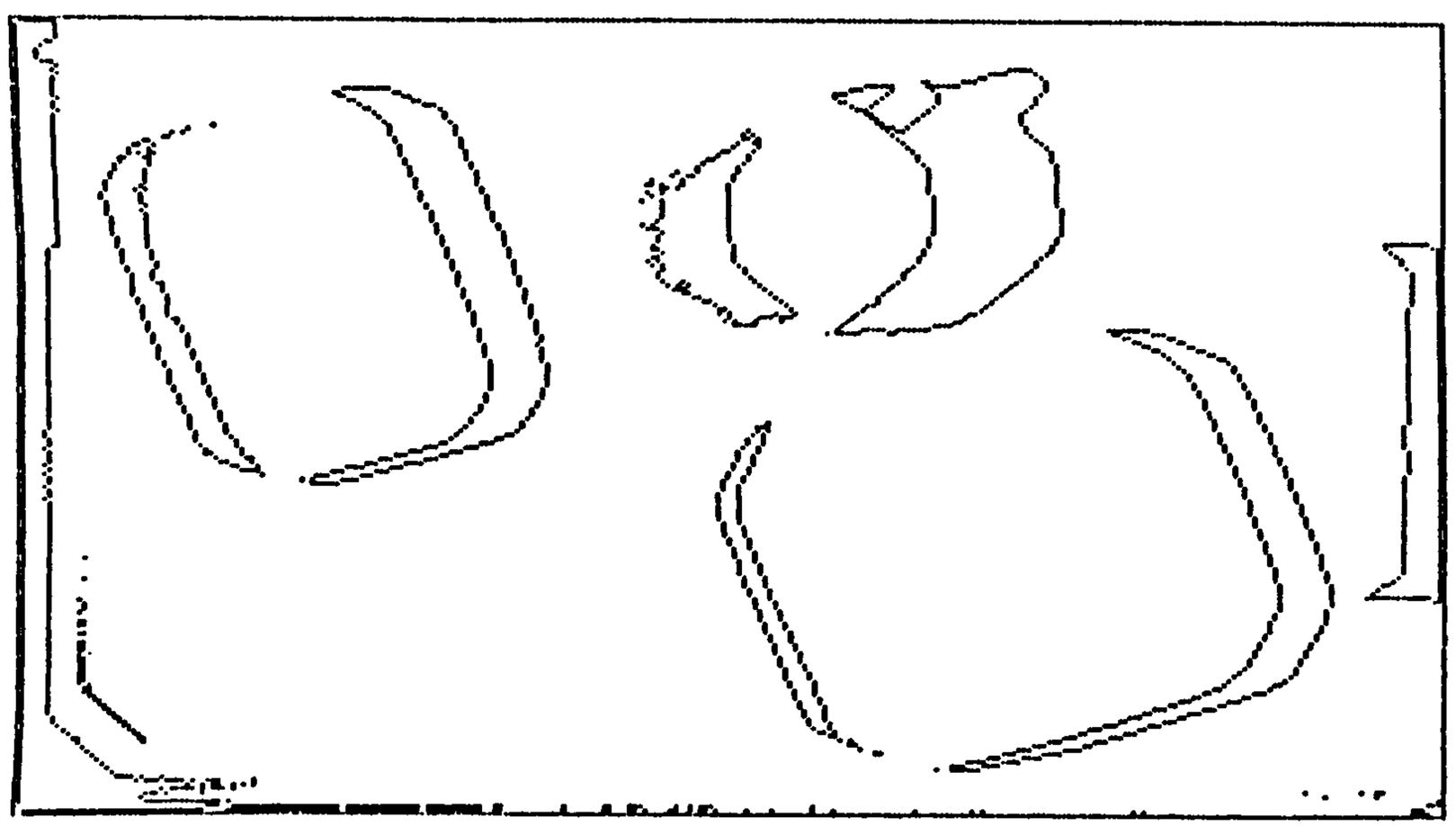


Figure 12.1c

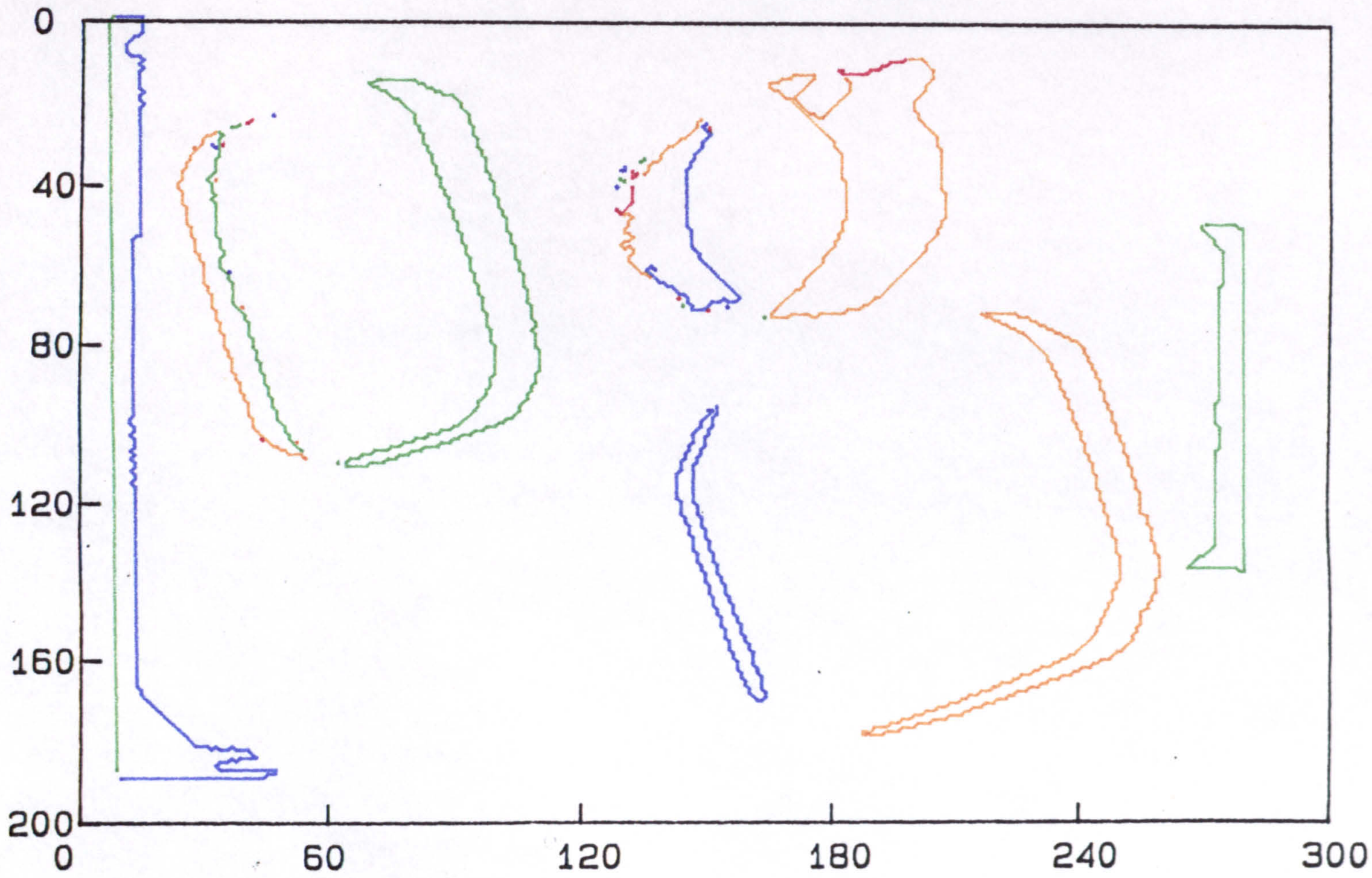
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	200	0	0	0	0	0	0	0	0	0
1	4	9	16	39	189	233	36	191	237	273	0	0	0	32	182	183	3	186	184
3	0	0	0	0	0	0	0	0	0	187	0	0	0	0	0	0	0	0	0
4	63	17	2	93	83	71	60	111	115	258	295	1140	76	-3	94	72	43	115	94
11	0	0	0	0	0	0	0	0	0	89	0	0	0	0	0	0	0	0	0
12	26	29	0	27	30	1	27	60	31	81	0	0	0	1	31	1	30	31	31
24	269	54	1	274	60	7	266	138	86	197	203	628	65	-3	84	7	78	85	84
31	214	75	0	248	136	66	184	181	143	311	360	1102	117	-30	106	69	78	147	110
33	137	121	126	141	105	126	137	115	116	150	113	43	296	0	-6	16	10	26	6

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.1c1



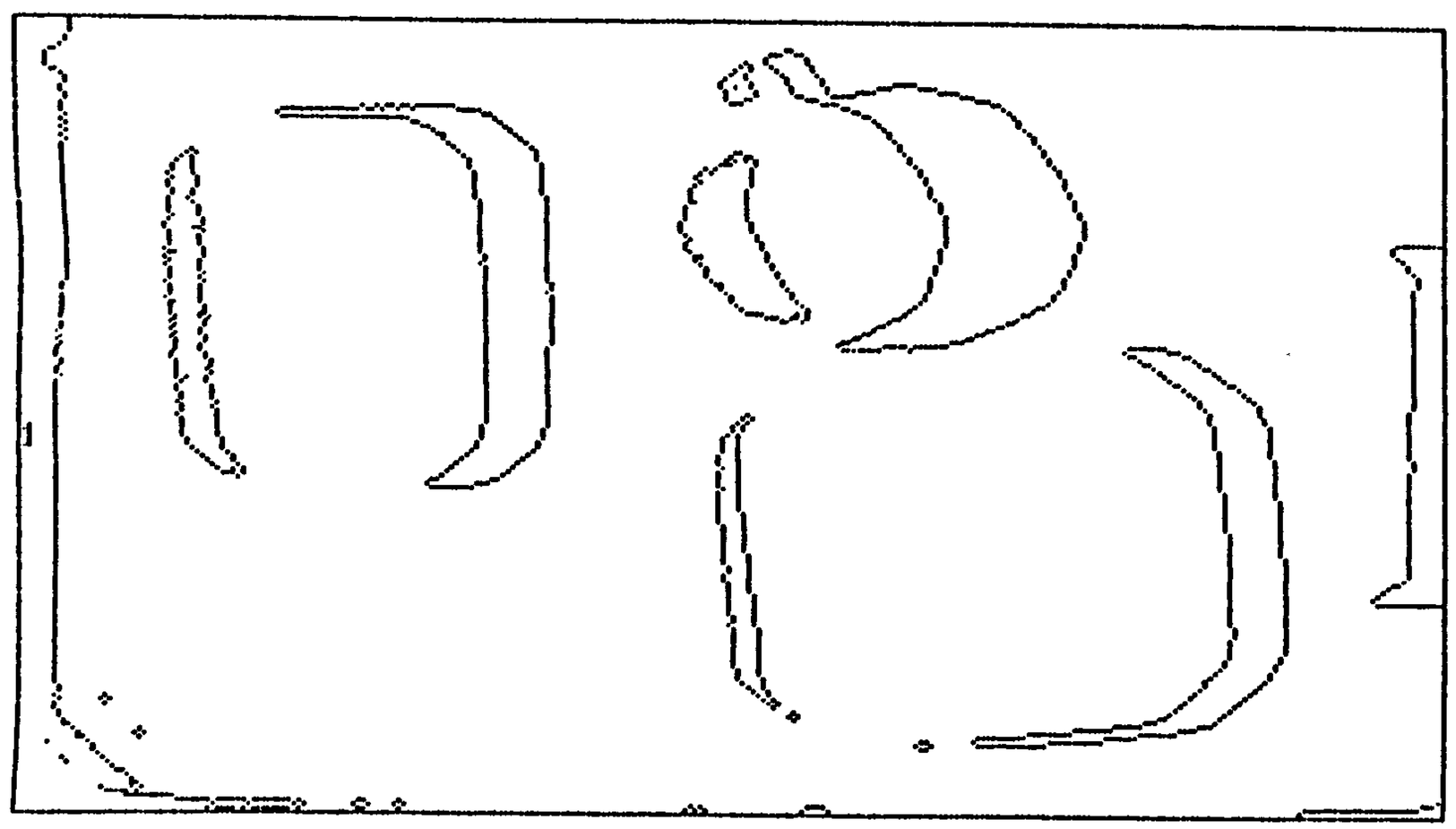


Figure 12.2a

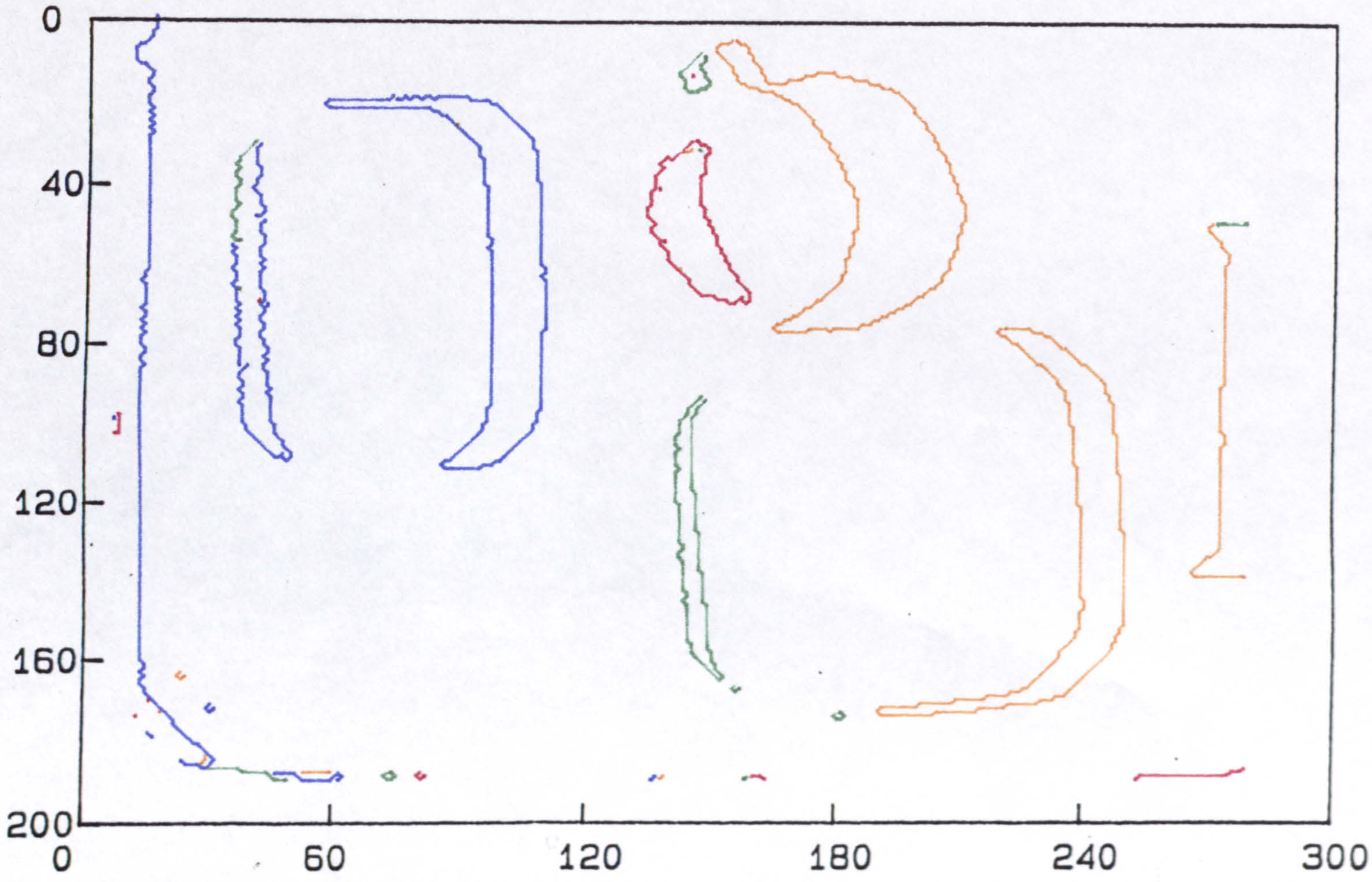
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	198	0	0	0	0	0	0	0	0	0
1	147	9	5	182	46	51	163	77	86	222	262	1749	39	16	68	50	36	86	69
4	51	22	17	92	56	78	82	111	133	263	290	1269	66	31	89	53	55	108	94
7	133	37	8	134	38	9	132	52	23	96	116	523	25	-1	15	1	14	15	15
12	35	54	3	43	109	58	42	111	60	88	0	0	0	7	57	55	2	57	57
13	270	54	2	275	60	8	267	138	87	102	0	0	0	-3	84	7	78	85	84
21	217	79	1	239	146	71	197	173	121	282	314	1174	83	-20	94	70	49	119	96
22	0	0	0	0	0	0	0	0	0	142	157	385	64	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.2a1



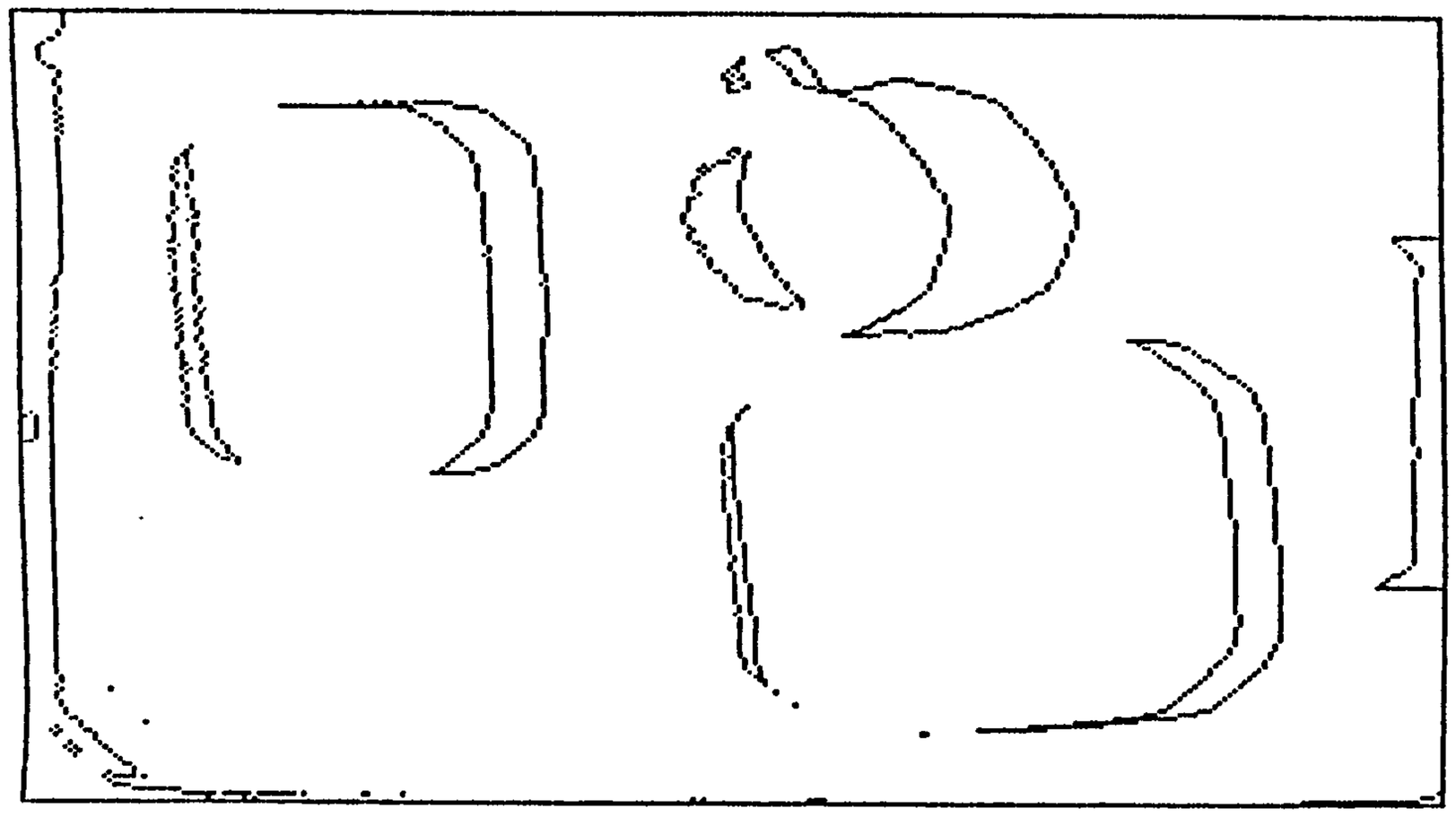


Figure 12.2b

Print of current image being worked on.

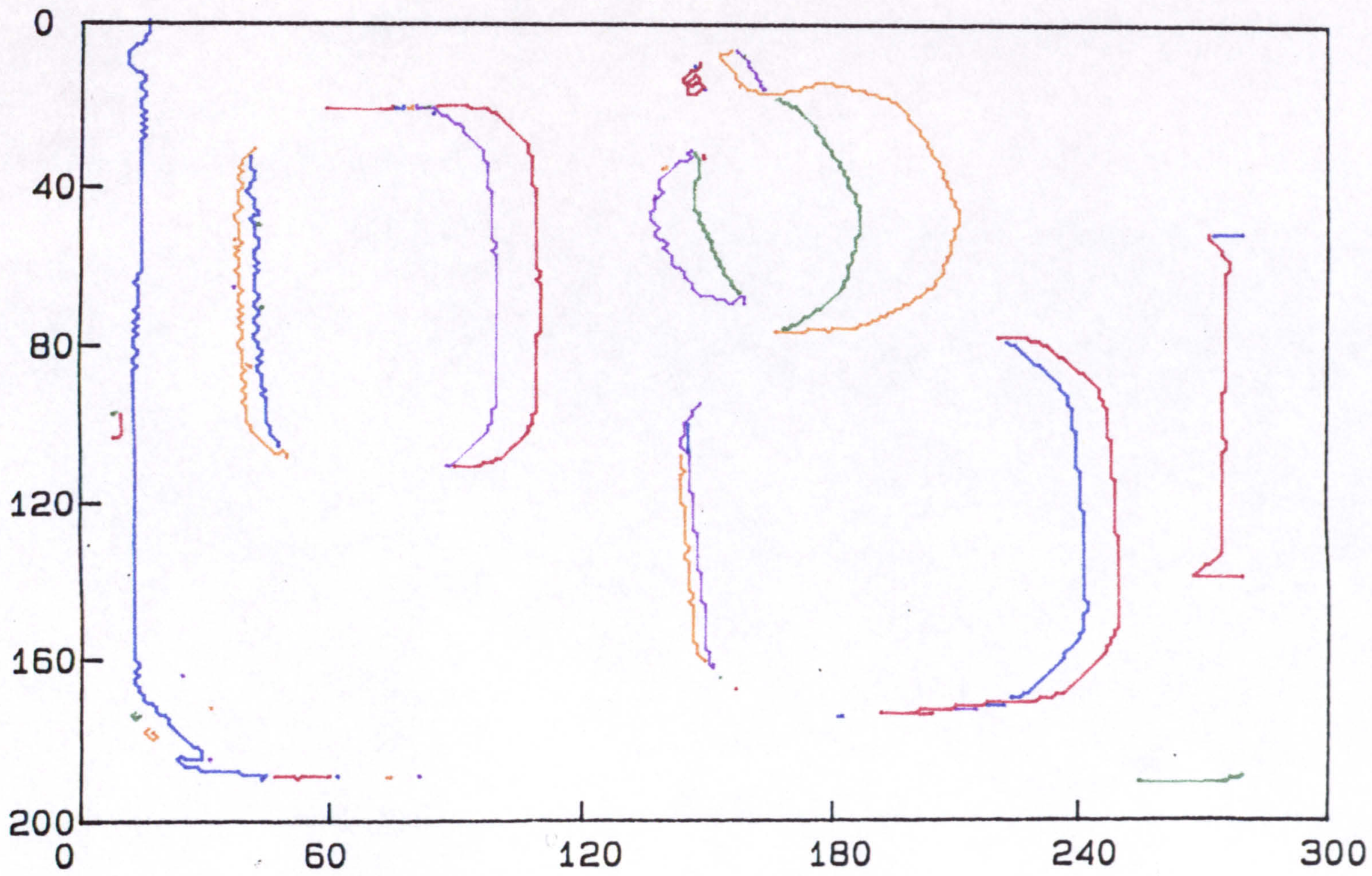
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	221	0	0	0	0	0	0	0	0	0
1	148	9	4	208	47	76	175	78	122	136	0	0	0	27	69	71	45	116	74
12	78	22	5	93	56	41	82	112	97	99	0	0	0	4	90	37	57	94	90
14	86	21	7	104	63	54	89	112	106	113	0	0	0	3	91	45	51	96	91
16	0	0	0	0	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0
24	271	54	1	276	60	7	268	138	86	100	0	0	0	-3	84	7	78	85	84
29	225	79	7	248	129	62	197	174	139	149	0	0	0	-28	95	55	68	123	99
30	220	80	0	240	146	68	223	170	93	96	0	0	0	3	90	68	29	97	90

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.2b1



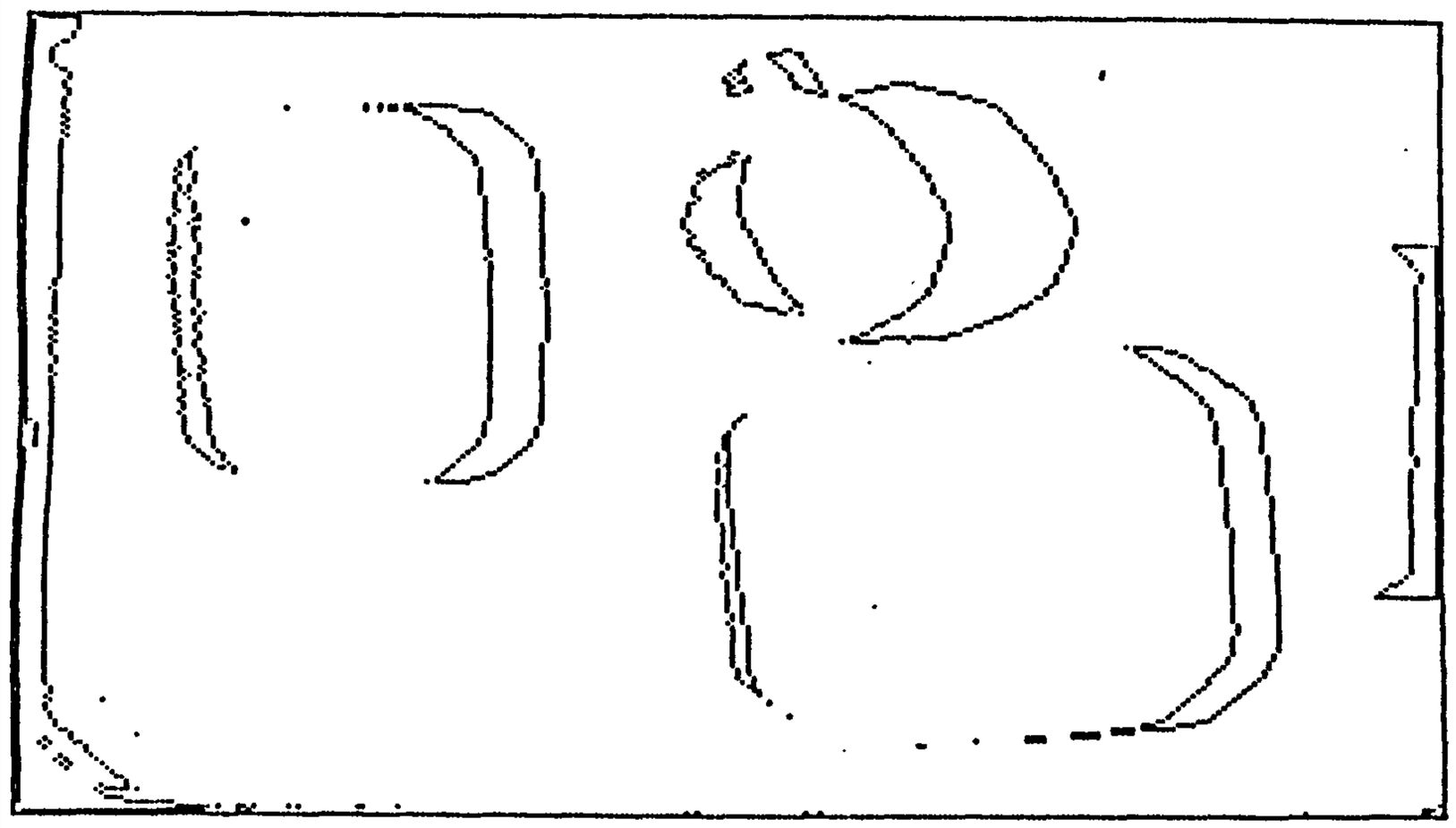


Figure 12.2c

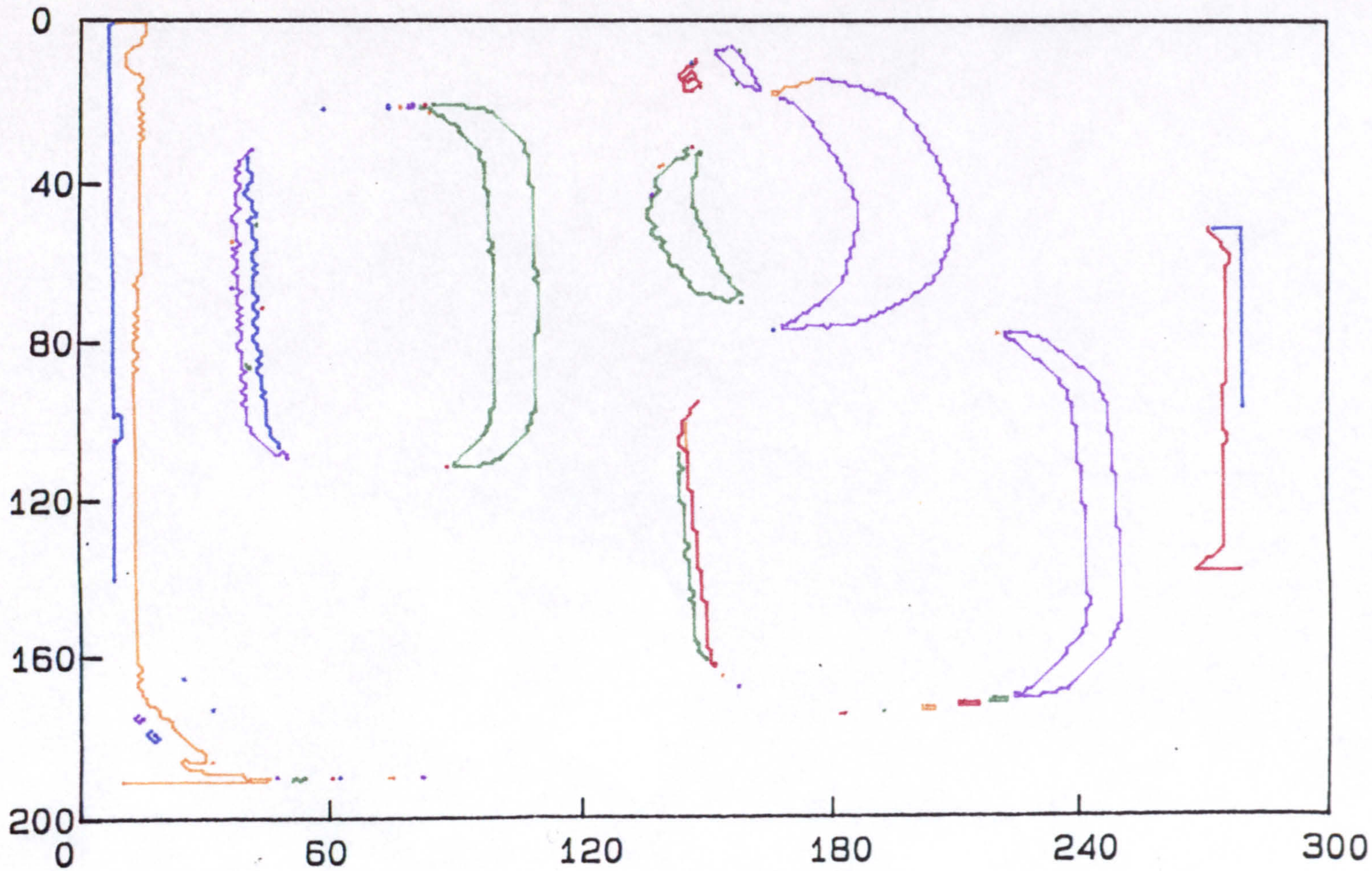
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	196	0	0	0	0	0	0	0	0	0
1	4	11	19	37	190	226	31	191	232	263	0	0	0	27	180	182	6	188	182
7	165	20	166	183	46	139	165	77	105	168	0	0	0	0	57	31	35	66	57
13	78	22	3	93	56	39	84	110	93	208	225	1002	50	6	88	37	54	91	88
17	0	0	0	0	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0
18	140	33	1	141	34	2	135	56	27	90	55	136	22	-5	23	1	22	23	23
24	271	54	1	276	60	7	268	138	86	99	0	0	0	-3	84	7	78	85	84
32	220	80	1	240	146	69	222	171	95	210	238	882	64	2	91	68	30	98	91

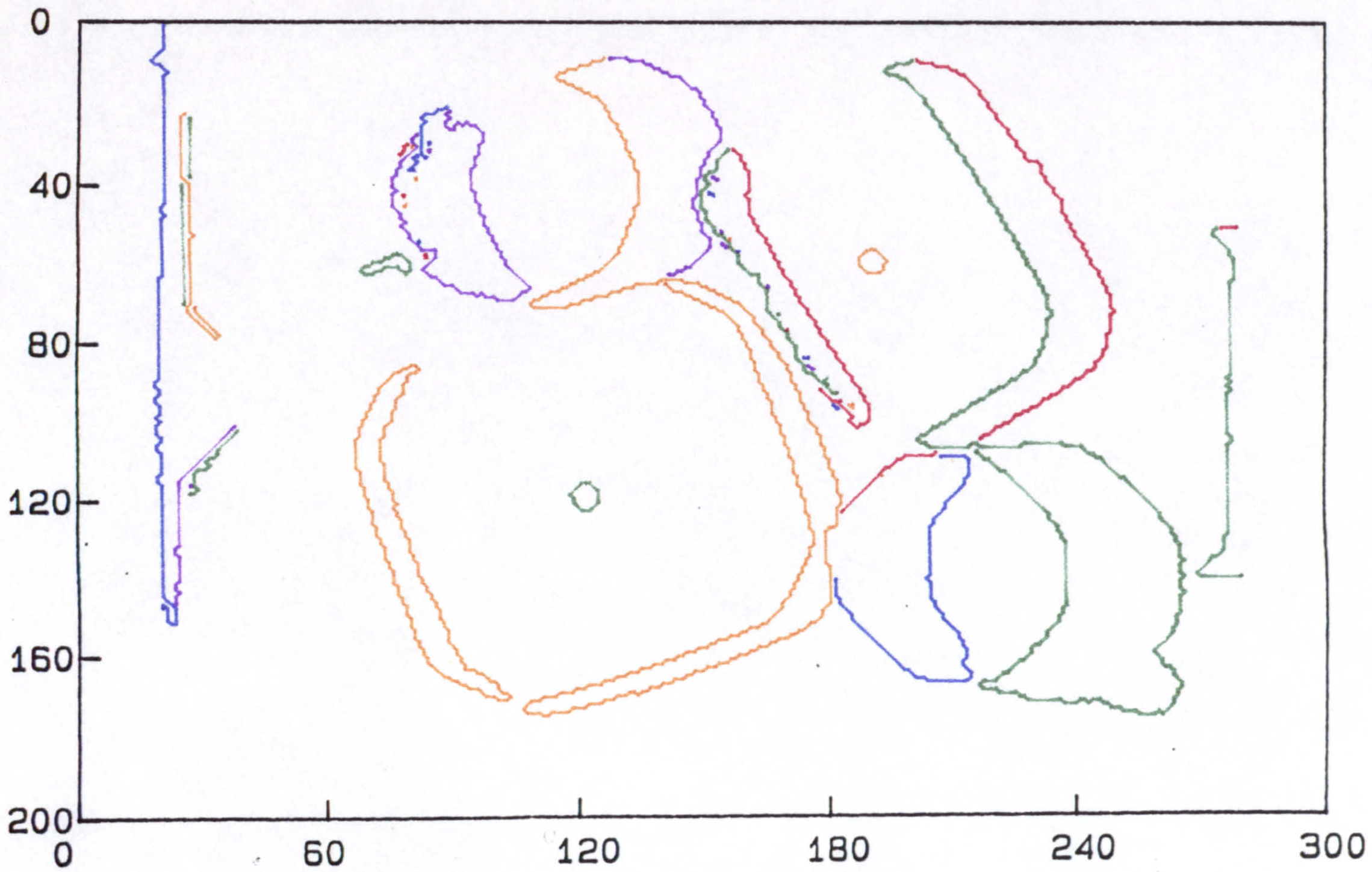
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.2c1



PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.3a1



PAGE NUMBERING AS ORIGINAL

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.3c1

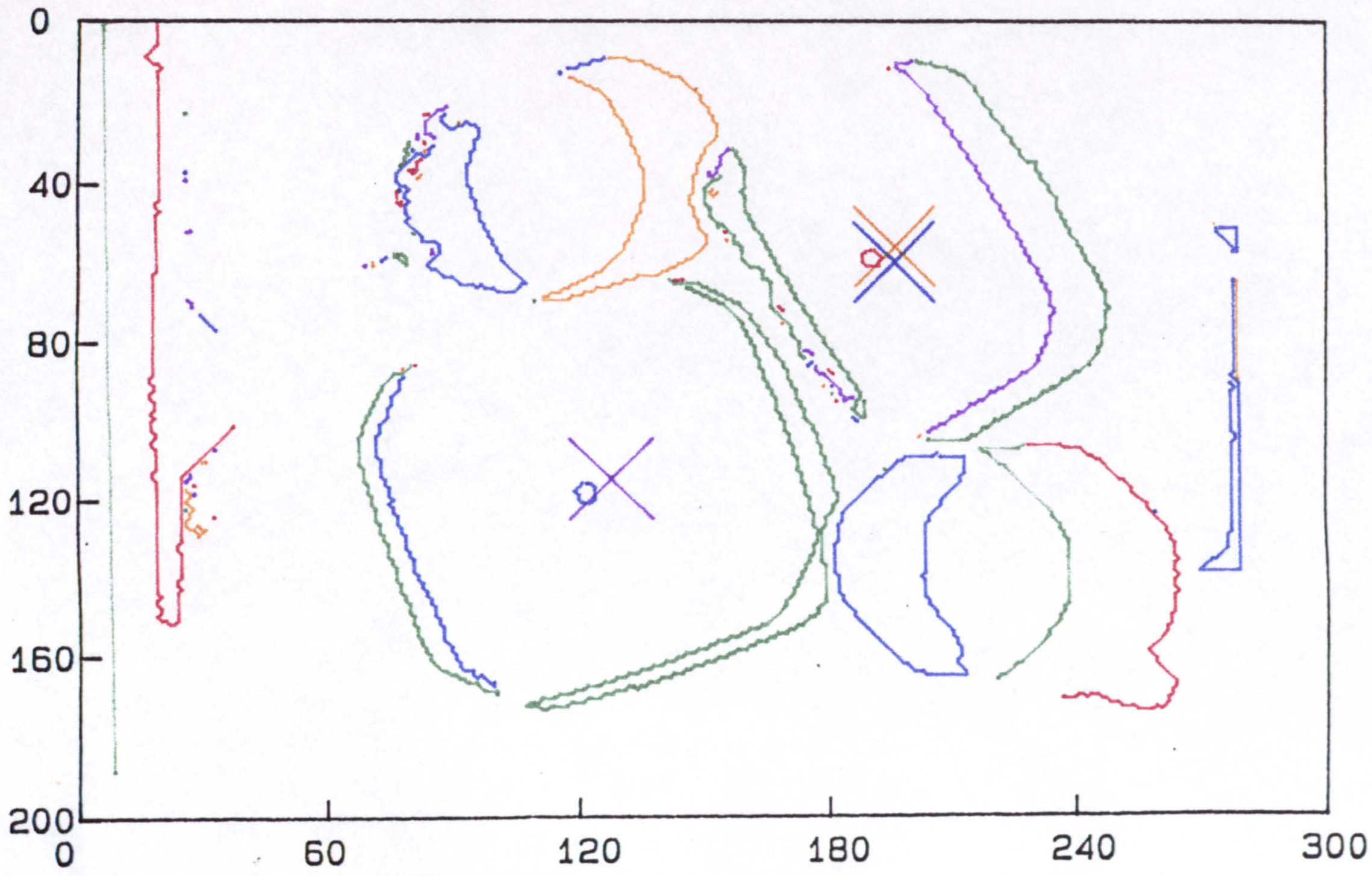


Figure 12.4a

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	205	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	212	256	1706	38	0	0	0	0	0	0
2	62	17	3	92	83	72	57	111	118	265	304	1403	65	-5	94	72	44	116	94
3	38	25	1	38	25	1	38	25	1	4	5	25	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	10	1	0	-1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	3	1	0	-1	0	0	0	0	0	0
7	145	26	0	154	71	45	145	75	56	68	0	0	0	0	49	45	9	54	49
8	15	43	16	15	43	16	15	43	16	175	203	730	56	0	0	0	0	0	0
9	26	28	0	26	28	0	26	28	0	3	1	30	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0
13	122	43	11	126	43	15	123	50	22	25	0	0	0	1	7	4	7	11	7
14	0	0	0	0	0	0	0	0	0	2	1	0	-1	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	2	2	0	-1	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	3	2	0	-1	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	2	2	0	-1	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0
20	124	51	0	125	52	1	125	53	2	7	0	0	0	1	2	1	1	2	2
21	0	0	0	0	0	0	0	0	0	2	2	51	0	0	0	0	0	0	0
22	268	54	2	273	60	8	265	138	87	104	0	0	0	-3	84	7	78	85	84
23	277	52	7	277	52	7	277	52	7	10	1	53	0	0	0	0	0	0	0

Figure 12.4a Continued.

24	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
25	123	58	1	123	58	1	123	58	1	19	0	0	0	0	0	0	0	0	0	
26	125	58	0	125	58	0	125	58	0	3	3	59	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
29	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
31	213	75	1	247	136	67	182	180	145	320	370	1422	96	-31	105	69	78	147	109	
32	0	0	0	0	0	0	0	0	0	93	0	0	0	0	0	0	0	0	0	
33	142	122	24	142	122	24	142	122	24	79	0	0	0	0	0	0	0	0	0	
34	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
35	11	131	1	11	131	1	11	131	1	4	5	131	0	0	0	0	0	0	0	
36	11	136	1	11	136	1	11	136	1	4	5	136	0	0	0	0	0	0	0	
37	10	161	52	25	175	29	0	0	0	63	0	0	0	-10	-161	20	176	196	161	
38	12	141	0	13	142	1	13	147	6	9	0	0	0	1	6	1	5	6	6	
39	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
40	39	183	2	41	185	4	39	188	7	22	0	0	0	0	5	2	3	5	5	
41	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	0	0	5	2	0	-1	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
45	252	186	1	252	186	1	252	186	1	4	5	186	0	0	0	0	0	0	0	
46	263	186	1	269	186	7	267	188	9	11	9	179	0	4	2	6	2	8	4	
47	0	0	0	0	0	0	0	0	0	7	2	0	-1	0	0	0	0	0	0	
48	256	187	1	256	187	1	256	187	1	4	5	187	0	0	0	0	0	0	0	
49	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	

Figure 12.4a Continued.

24	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
25	123	58	1	123	58	1	123	58	1	19	0	0	0	0	0	0	0	0	0	
26	125	58	0	125	58	0	125	58	0	3	3	59	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
29	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
31	213	75	1	247	136	67	182	180	145	320	370	1422	96	-31	105	69	78	147	109	
32	0	0	0	0	0	0	0	0	0	93	0	0	0	0	0	0	0	0	0	
33	142	122	24	142	122	24	142	122	24	79	0	0	0	0	0	0	0	0	0	
34	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
35	11	131	1	11	131	1	11	131	1	4	5	131	0	0	0	0	0	0	0	
36	11	136	1	11	136	1	11	136	1	4	5	136	0	0	0	0	0	0	0	
37	10	161	52	25	175	29	0	0	0	63	0	0	0	-10	-161	20	176	196	161	
38	12	141	0	13	142	1	13	147	6	9	0	0	0	1	6	1	5	6	6	
39	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
40	39	183	2	41	185	4	39	188	7	22	0	0	0	0	5	2	3	5	5	
41	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	0	0	5	2	0	-1	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	
45	252	186	1	252	186	1	252	186	1	4	5	186	0	0	0	0	0	0	0	
46	263	186	1	269	186	7	267	188	9	11	9	179	0	4	2	6	2	8	4	
47	0	0	0	0	0	0	0	0	0	7	2	0	-1	0	0	0	0	0	0	
48	256	187	1	256	187	1	256	187	1	4	5	187	0	0	0	0	0	0	0	
49	0	0	0	0	0	0	0	0	0	1	1	0	-1	0	0	0	0	0	0	

Figure 12.4a Continued.

50	0	0	0	0	0	0	0	0	0	3	1	187	0	0	0	0	0	0	0	0
51	219	190	60	279	187	0	0	0	0	105	0	0	0	-219	-190	60	335	395	289	
52	122	189	80	122	189	80	122	189	80	83	0	0	0	0	0	0	0	0	0	
53	0	0	0	0	0	0	0	0	0	3	1	0	-1	0	0	0	0	0	0	
54	0	0	0	0	0	0	0	0	0	3	1	0	-1	0	0	0	0	0	0	
55	0	0	0	0	0	0	0	0	0	2	1	0	-1	0	0	0	0	0	0	
56	136	189	1	136	189	1	136	189	1	4	1	191	0	0	0	0	0	0	0	
57	0	0	0	0	0	0	0	0	0	10	1	0	-1	0	0	0	0	0	0	
58	0	0	0	0	0	0	0	0	0	2	1	0	-1	0	0	0	0	0	0	
59	160	190	3	160	190	3	160	190	3	5	1	191	0	0	0	0	0	0	0	

Figure 12.4c

TABLE of IMAGE FEATURES

Sample	min i	min j	min pid	mid i	mid j	mid pid	max i	max j	max pid	Bound	Perim	Area	Compact	mxx-mnx	mxy-mny	d(mn-md)	d(mx-md)	Total	d(mn-mx)
0	0	0	0	0	0	0	0	0	0	205	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	212	256	1706	38	0	0	0	0	0	0
2	62	17	3	92	83	72	57	111	118	265	304	1403	65	-5	94	72	44	116	94
7	145	26	0	154	71	45	145	75	56	68	0	0	0	0	49	45	9	54	49
8	15	43	16	15	43	16	15	43	16	175	203	730	56	0	0	0	0	0	0
22	268	54	2	273	60	8	265	138	87	104	0	0	0	-3	84	7	78	85	84
31	213	75	1	247	136	67	182	180	145	320	370	1422	96	-31	105	69	78	147	109
32	0	0	0	0	0	0	0	0	0	93	0	0	0	0	0	0	0	0	0
33	142	122	24	142	122	24	142	122	24	79	0	0	0	0	0	0	0	0	0
37	10	161	52	25	175	29	0	0	0	63	0	0	0	-10	-161	20	176	196	161
51	219	190	60	279	187	0	0	0	0	105	0	0	0	-219	-190	60	335	395	289
52	122	189	80	122	189	80	122	189	80	83	0	0	0	0	0	0	0	0	0

```
**************************************************
*                                                *
*      PROGRAM FOR DISK OR APPLE IMAGE INPUT     *
*        to CREATE DISK FEATURE-DATA FILES       *
*                                                *
**************************************************
```

Figure 12.5

```
******************************************************************
*                                                                *
*                     PROGRAM SET-UP PHASE                       *
*                                                                *
******************************************************************
```

Program Parameter Set-up.

Enter boundary outline-length threshold parameter:
Any line below 50 boundary points is discarded in subsequent processing stages.

Program Display Option Set-up.

Are the images to be read in from disk or from the Apple?

Enter 1 if the images are to be read in from disk drive 0.
Enter 2 if the images are to be read in from the Apple.

The images are to be received from the Apple.
Do you want to output feature data to DUET?

Is a dot-matrix printout of initial image required?
Is a dot-matrix printout of the thinned/edge-extracted outline required.
A dot-matrix printout of edge-extracted image will be produced.

Is a printout of the feature-table data required?
A feature-table printout will be provided.

Is a presentation of the chain-coded outlines required on the DUET?

Is end-of-plot sequence marker to be sent for each complete plot transferred.

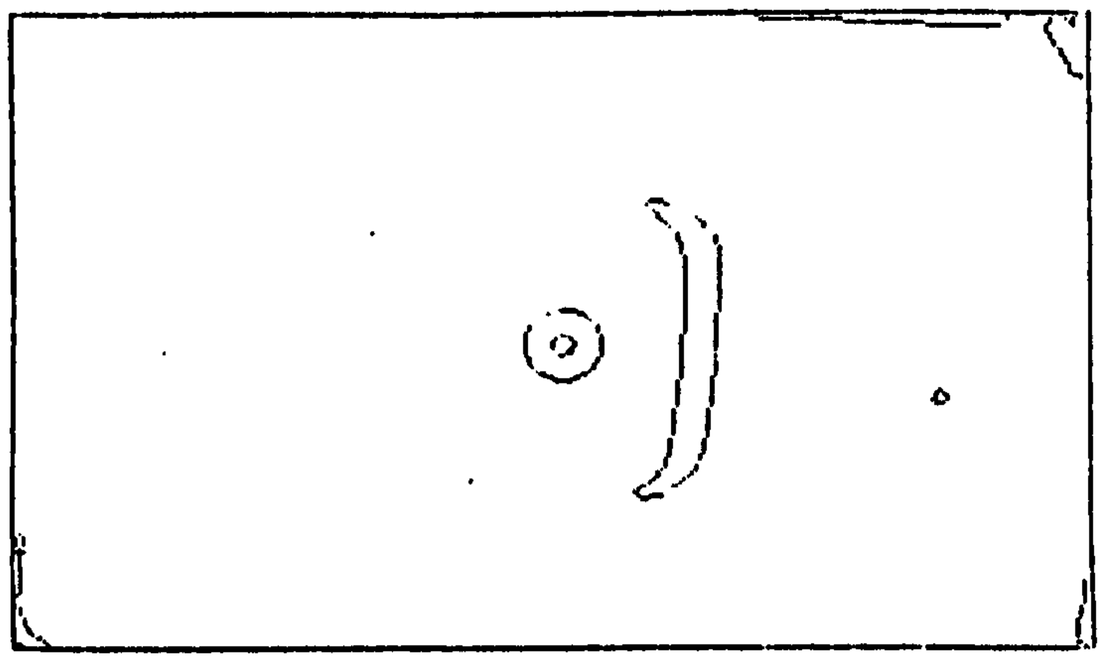
```
**************************************************
*                                                *
*          APPLE IMAGE INPUT PARAMETERS STAGE    *
*                                                *
**************************************************
```

Enter the number of file images to be read in from the APPLE.
Accessing 10 images from the APPLE.

Image received from Apple.

Expansion of bits into bytes completed.

Edge extraction completed.



Print of current image being worked on.

....
Chain coding sequence completed.

Test for closed outlines completed.

Artificial end of boundary generation completed.

Closed boundary perimeter calculation completed.
Closed boundary area calculation completed.
Closed boundary compaction-factor calculation completed.

Open line feature point derivation completed.

Open line feature calculation completed.
Centre marker centre calculated. x= 143, y= 100.
Centre circle coordinates calculated.
Calibration circle found. n= 4.
minds=2420, midds=1808, maxds=2290.

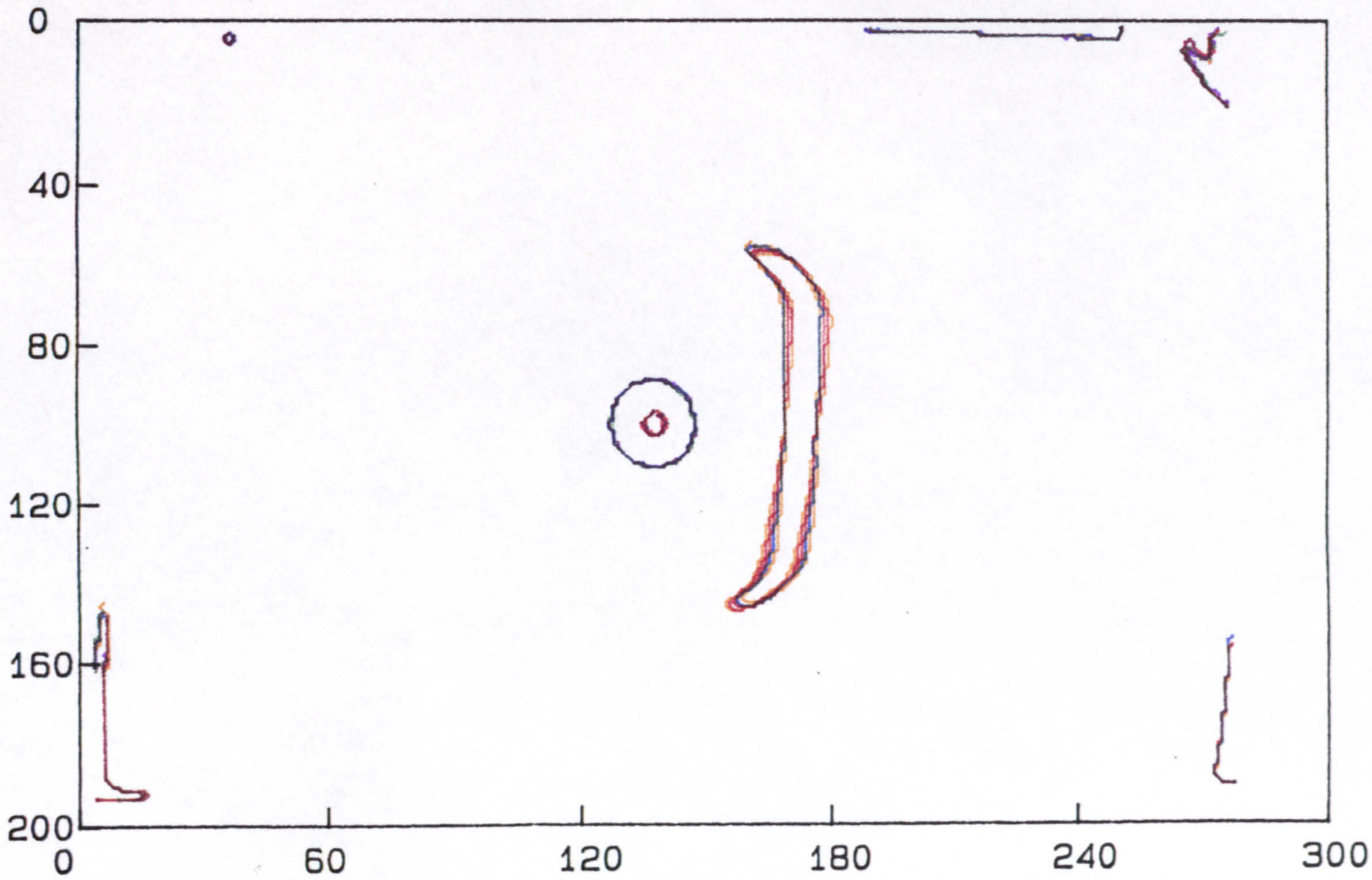
Distances to centre circle calculated.
Values being stored for output are:- 1,215,800,57,89,87,10,18,71,49,42,47.
Output feature arranging routine finished.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	67	0	0	0	0	0	0	0
3	165	56	1	175	72	17	164	143	89	197	215	800	57	18	71	87	89
4	0	0	0	0	0	0	0	0	0	60	69	441	10	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.6



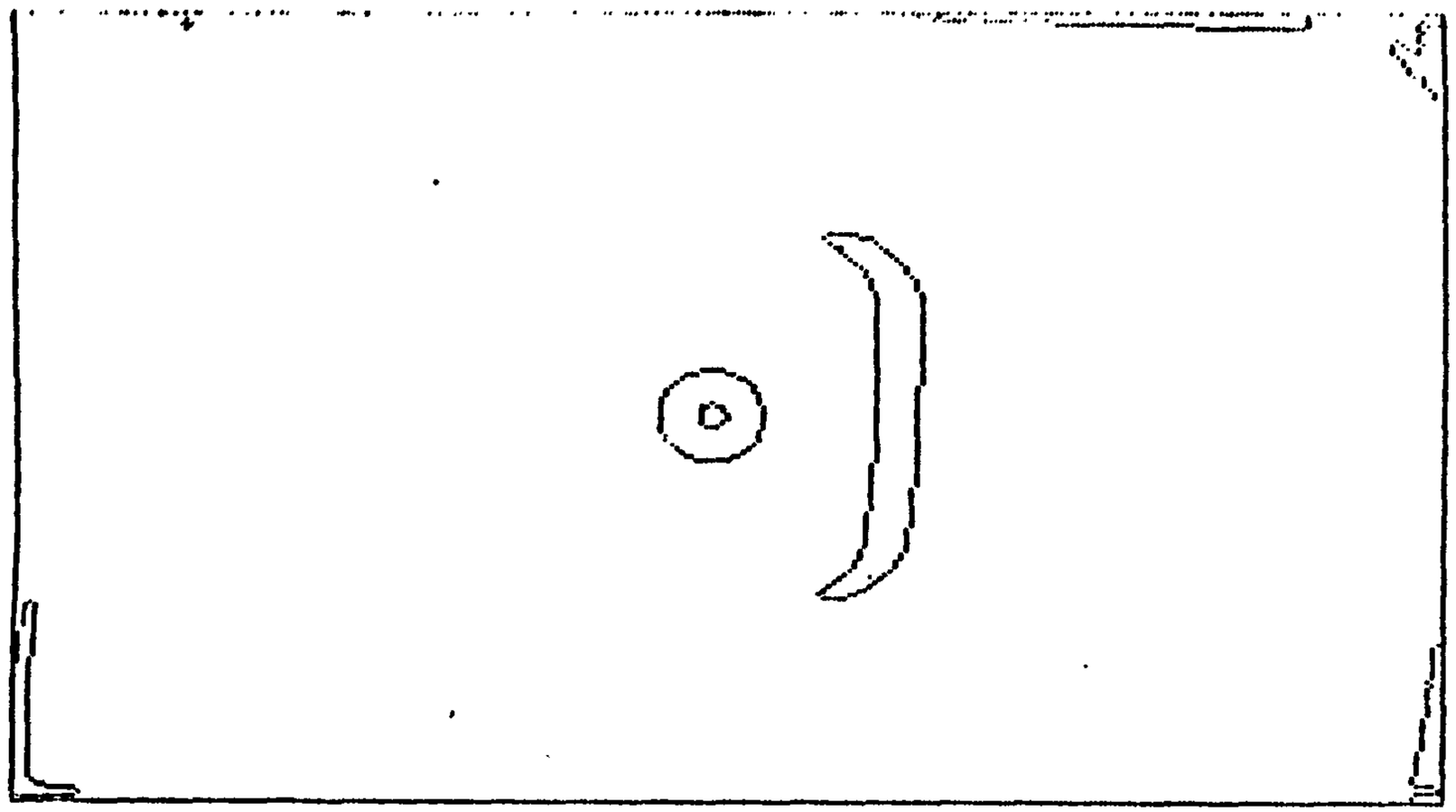


Figure 12.7a

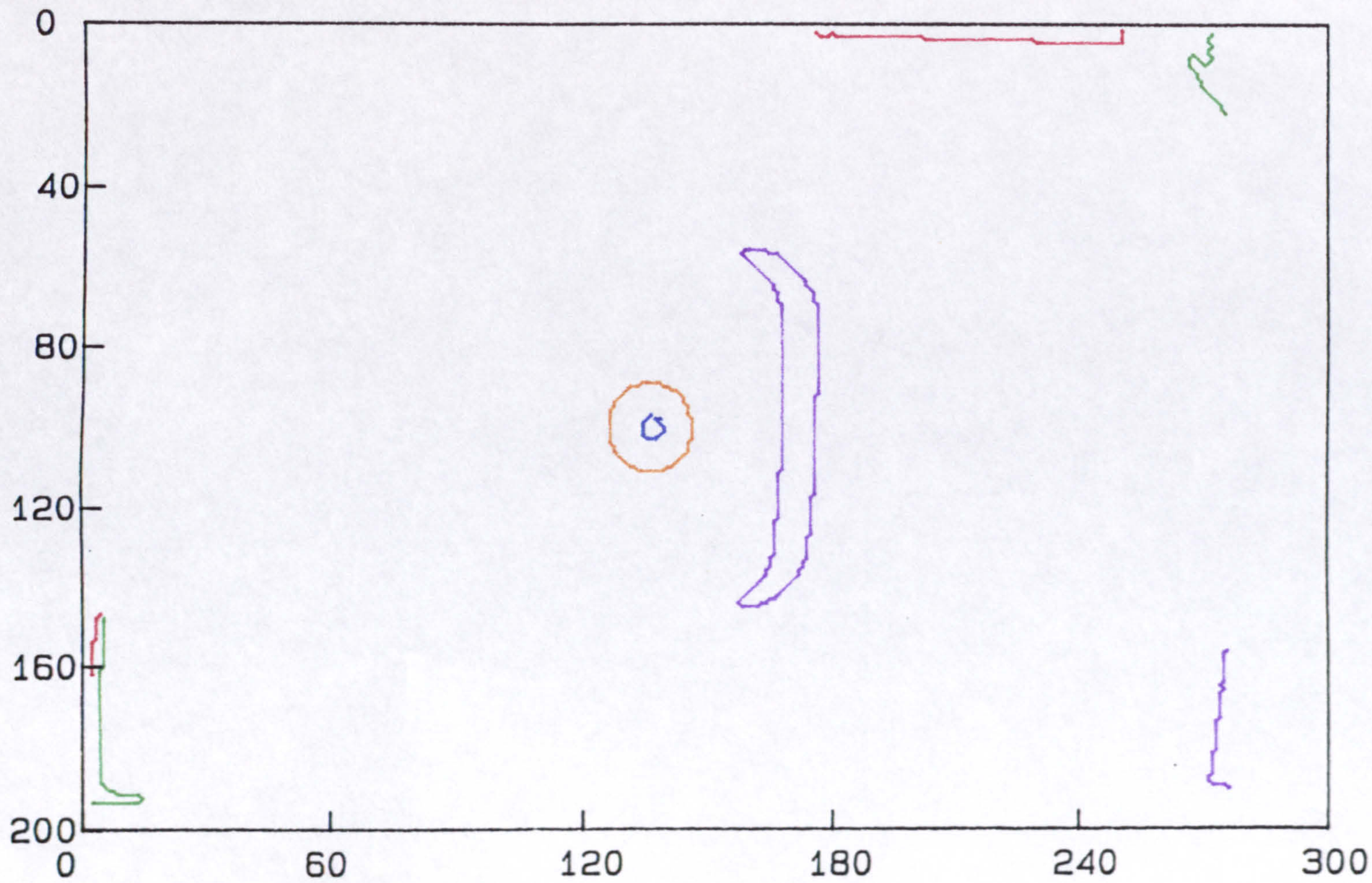
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	80	0.	0	0	0	0	0	0
7	158	55	1	168	68	14	159	140	86	191	206	748	56	16	72	85	88
9	0	0	0	0	0	0	0	0	0	60	69	440	10	0	0	0	0
13	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.7b



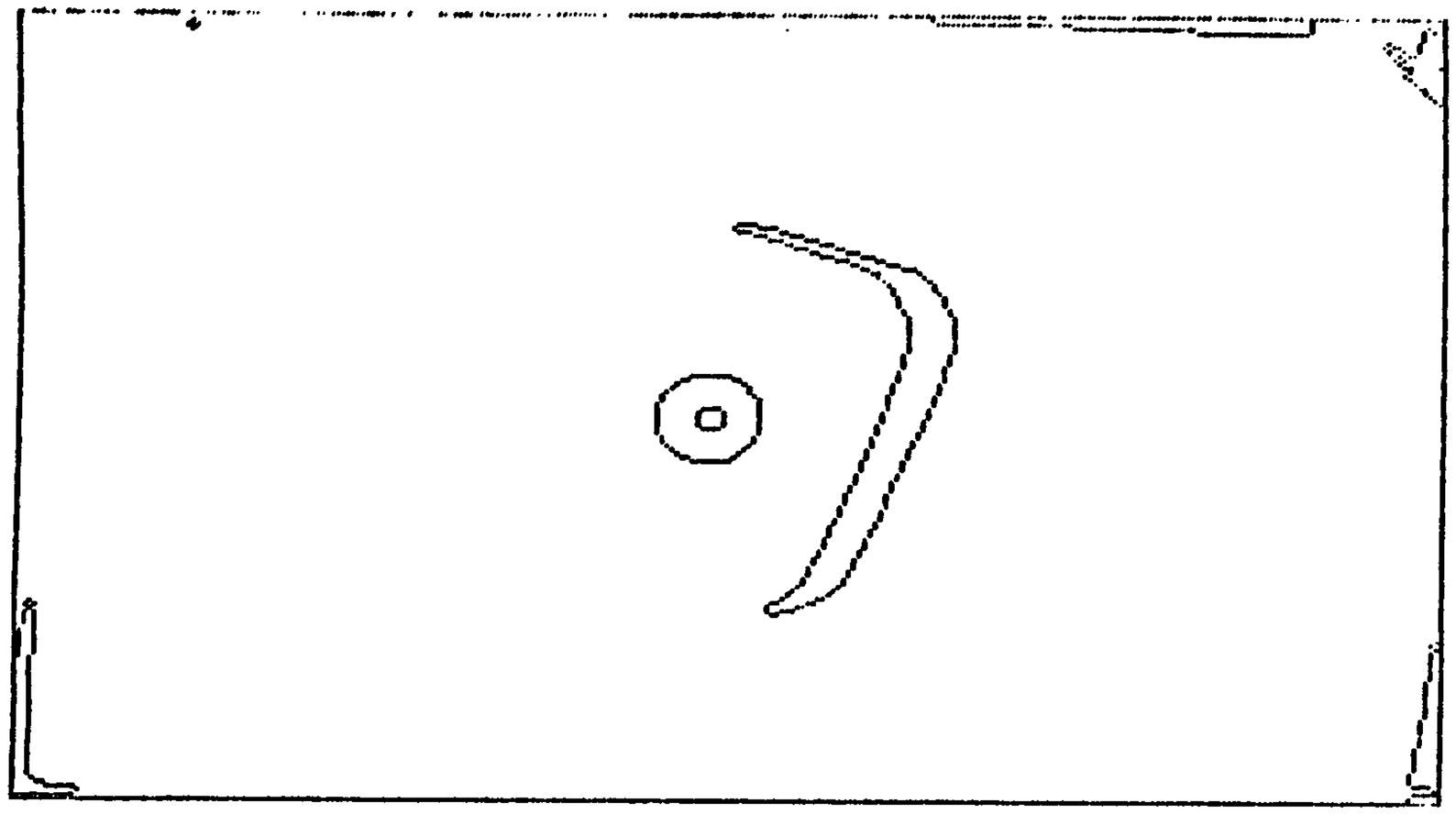


Figure 12.8a

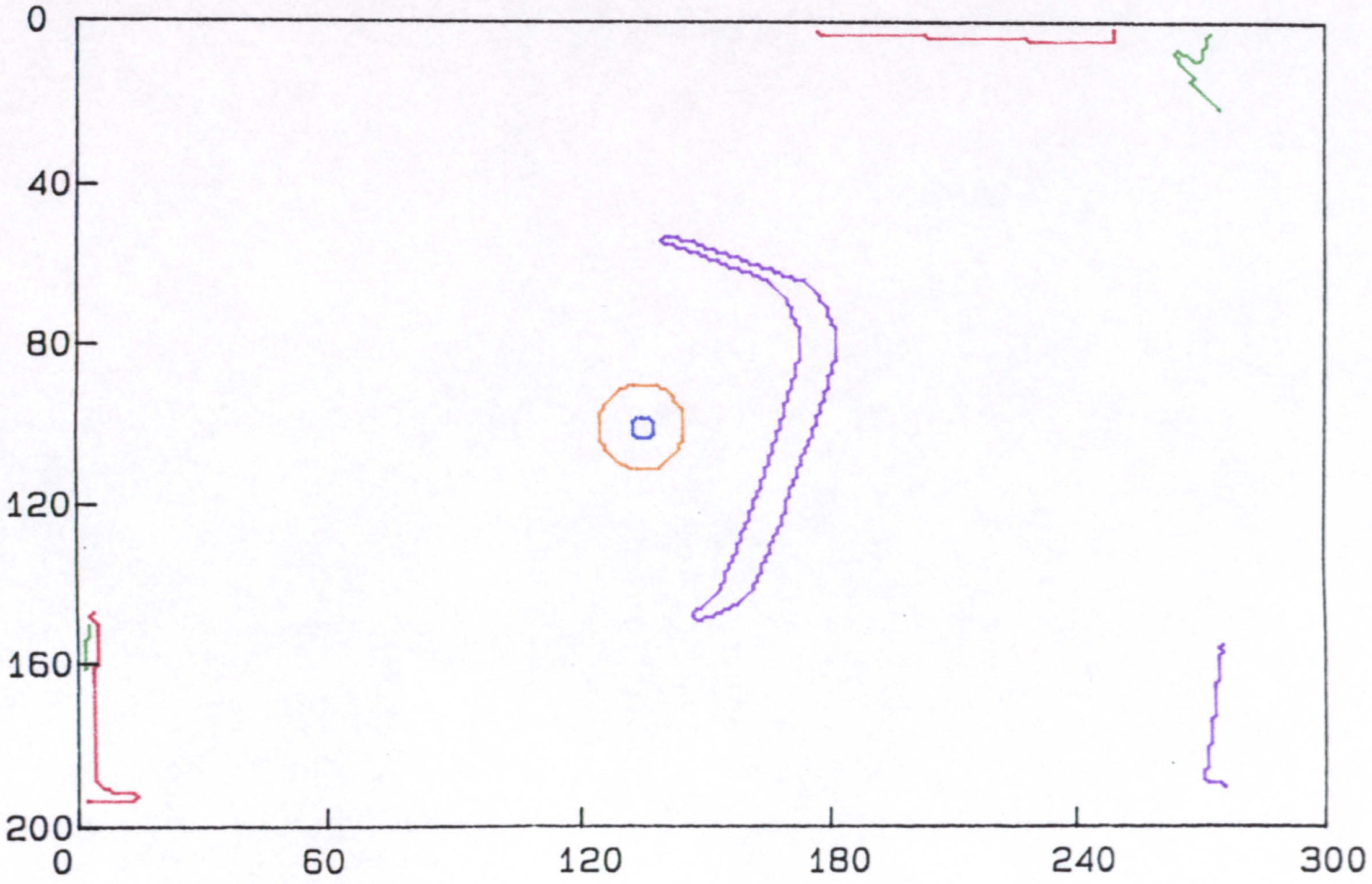
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0
9	140	52	1	174	74	39	149	144	109	238	272	785	94	40	74	92	114
11	0	0	0	0	0	0	0	0	0	59	68	433	10	0	0	0	0
13	0	0	0	0	0	0	0	0	0	71	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.8b



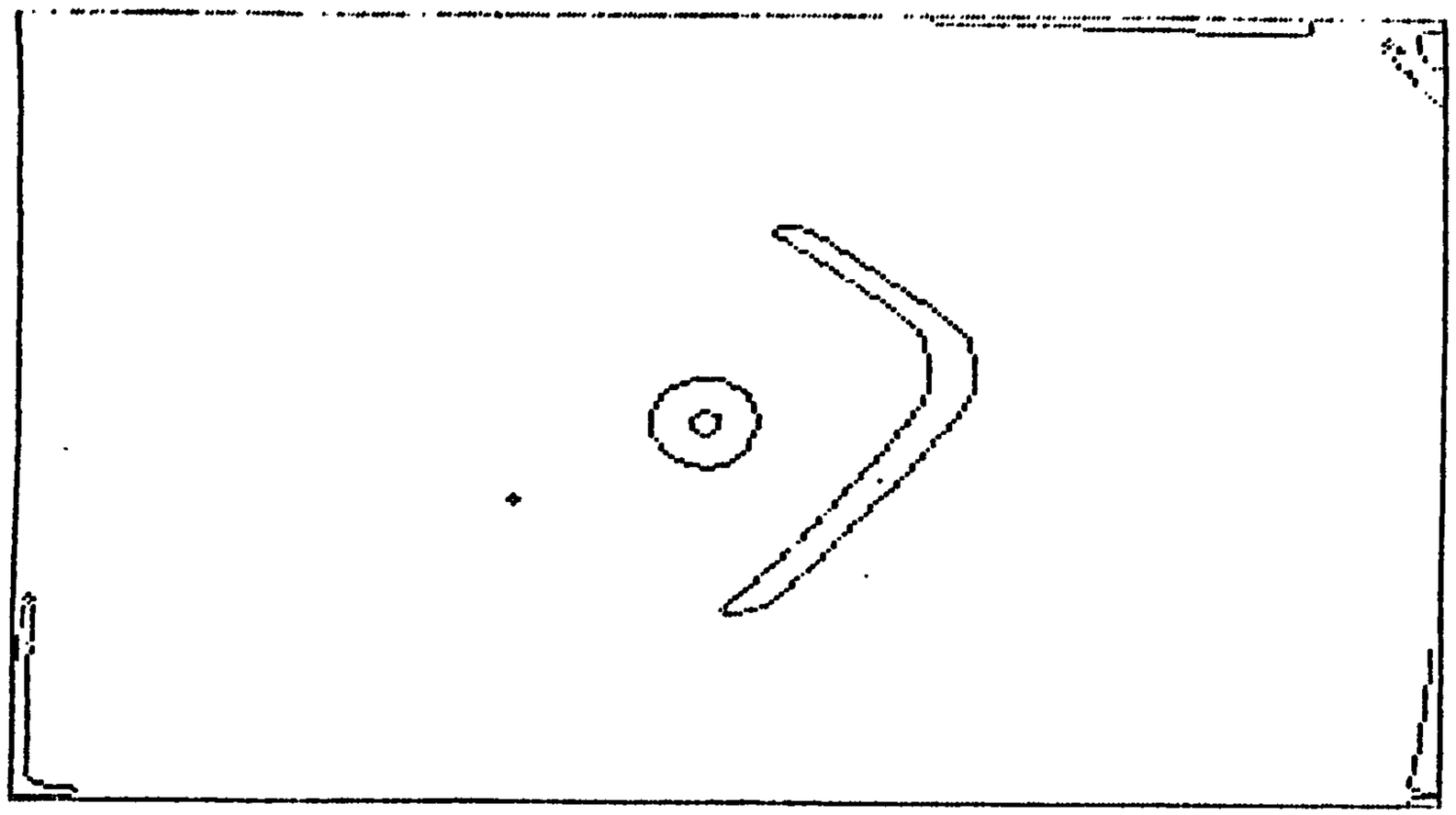


Figure 12.9a

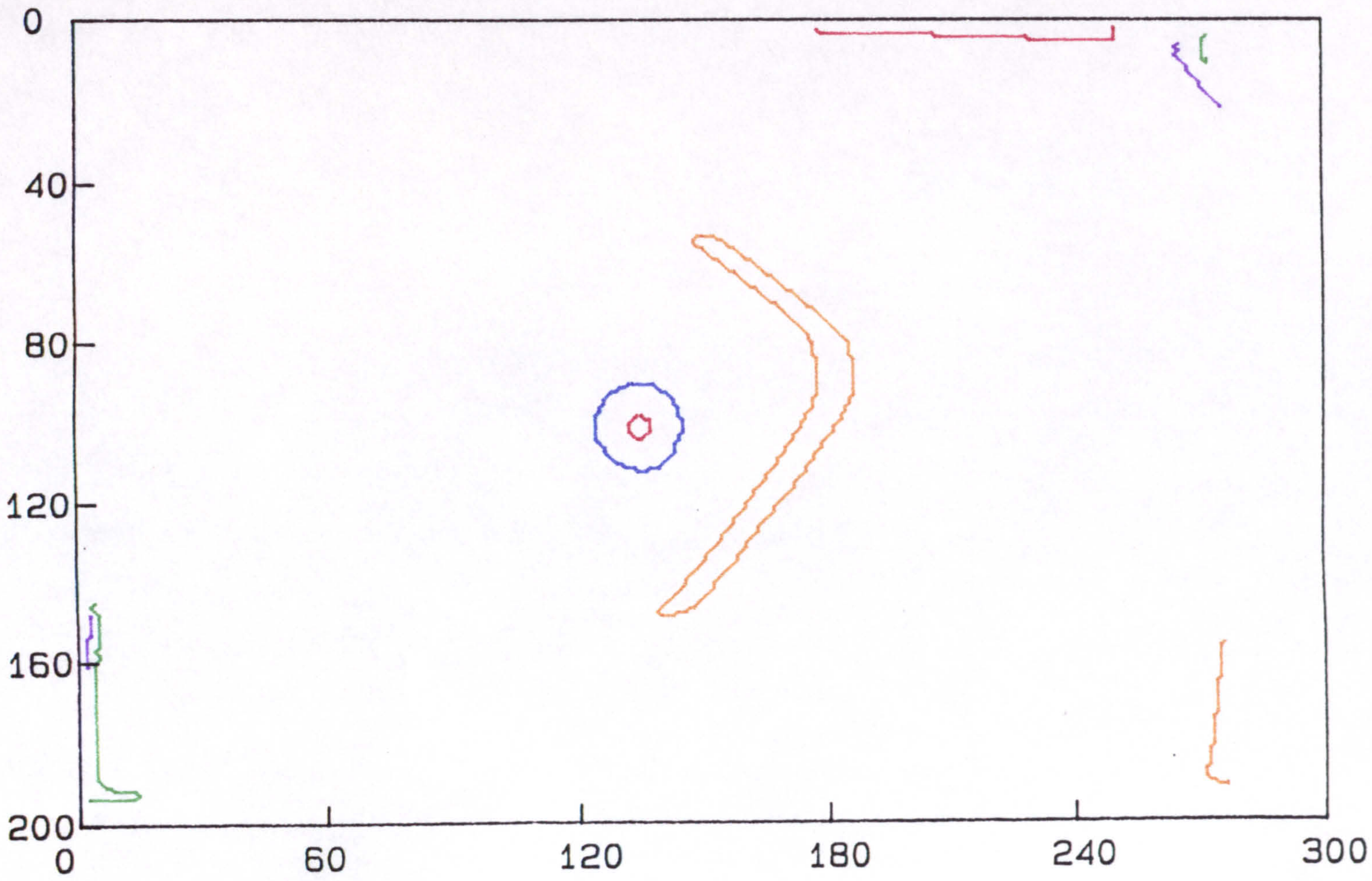
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0
9	148	53	2	178	82	36	140	143	97	211	262	766	89	41	71	90	112
10	0	0	0	0	0	0	0	0	0	60	70	440	11	0	0	0	0
13	0	0	0	0	0	0	0	0	0	73	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.9b



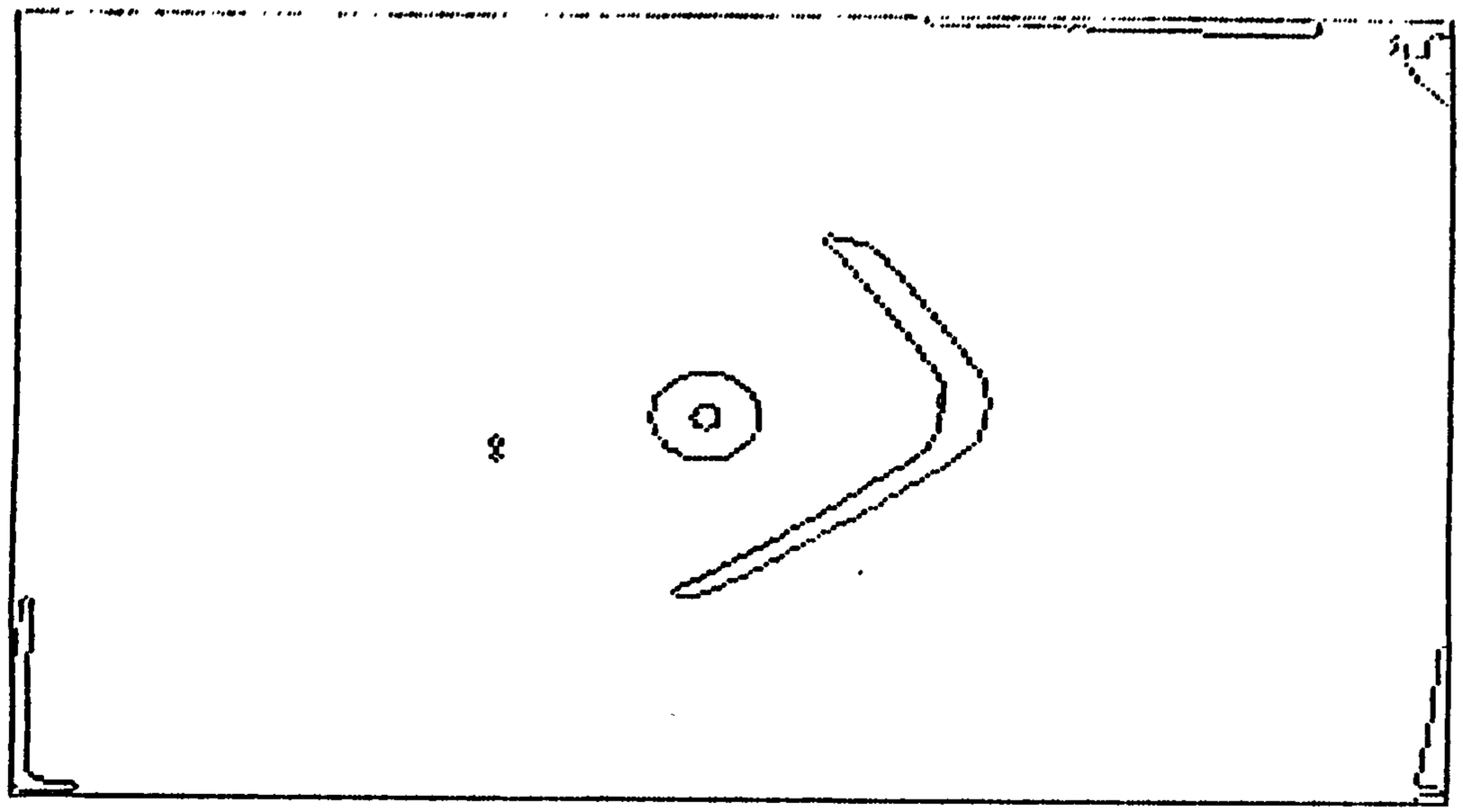


Figure 12.10a

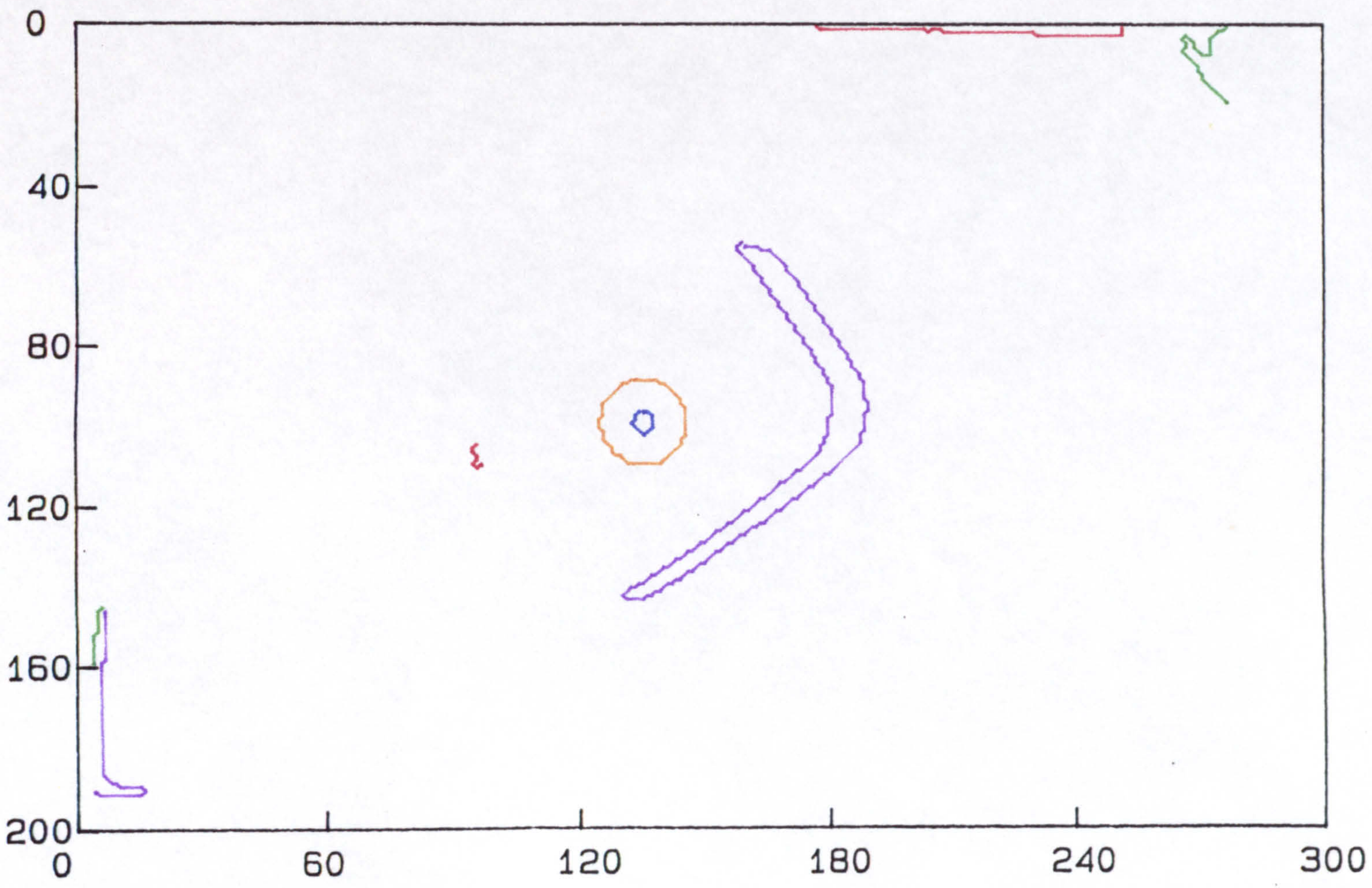
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0
6	157	56	2	180	90	36	130	140	99	215	263	704	98	41	70	88	111
7	0	0	0	0	0	0	0	0	0	59	69	436	10	0	0	0	0
14	0	0	0	0	0	0	0	0	0	103	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.10b



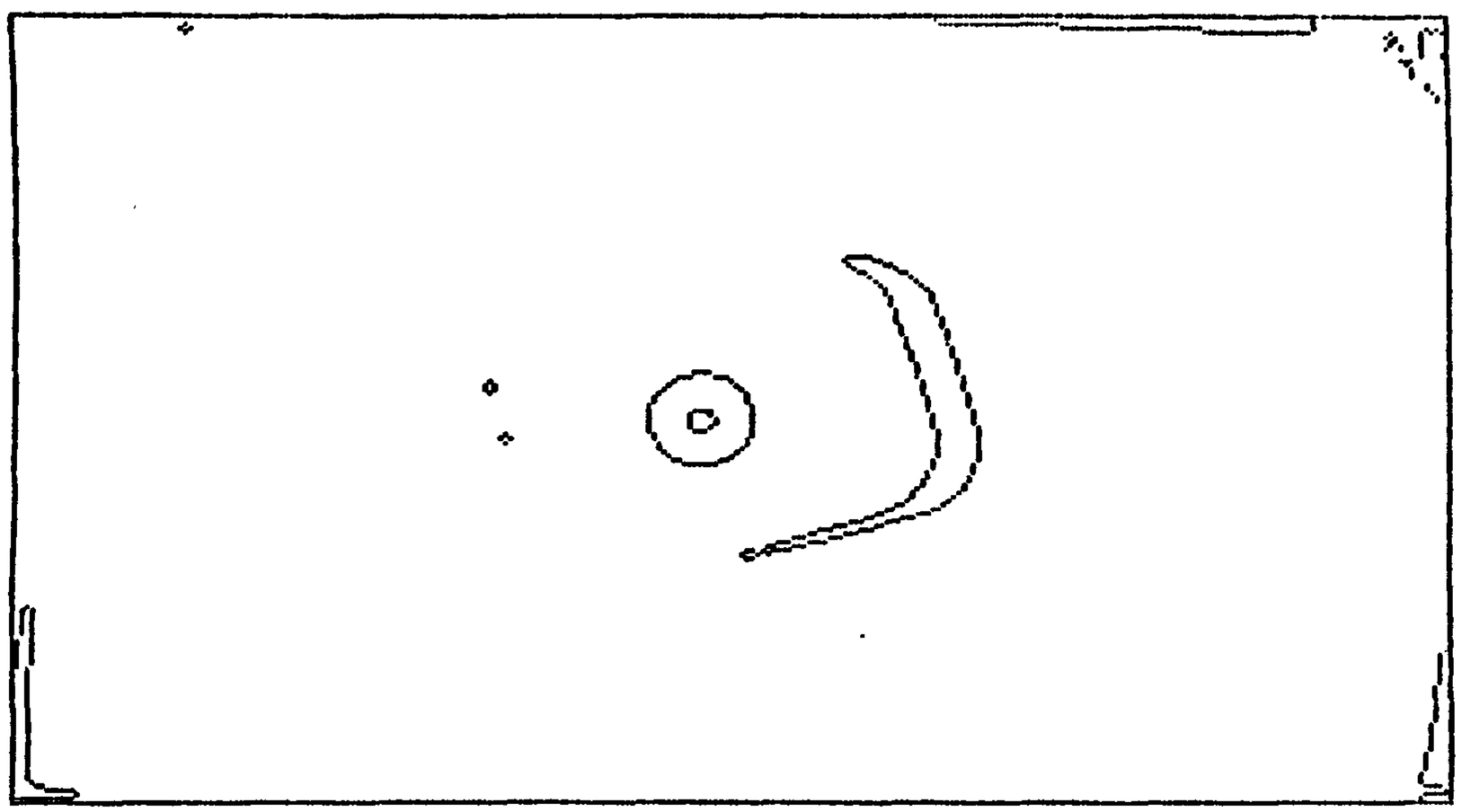


Figure 12.11a

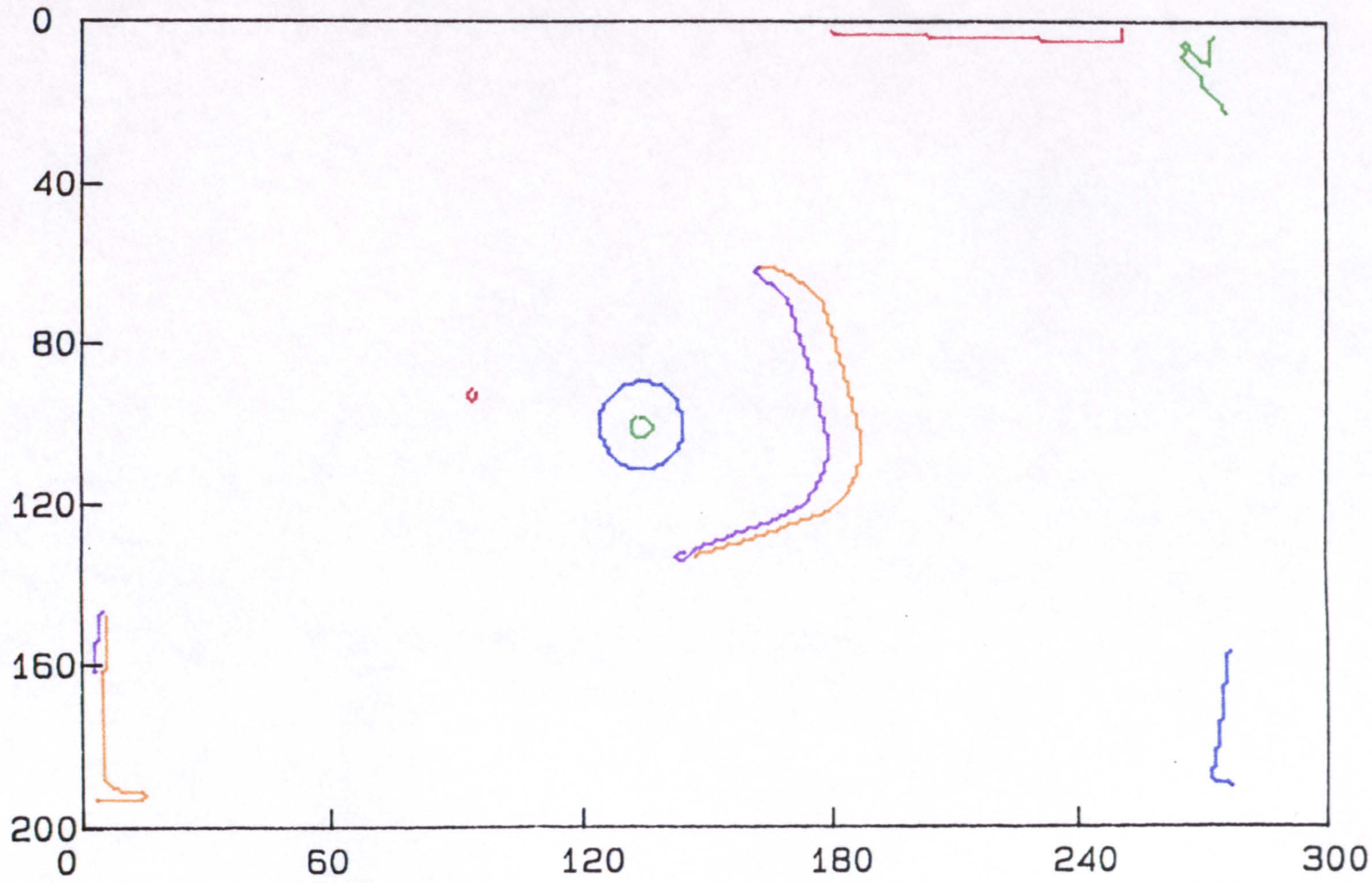
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	95	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	101	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	59	69	435	10	0	0	0	0
17	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.11b



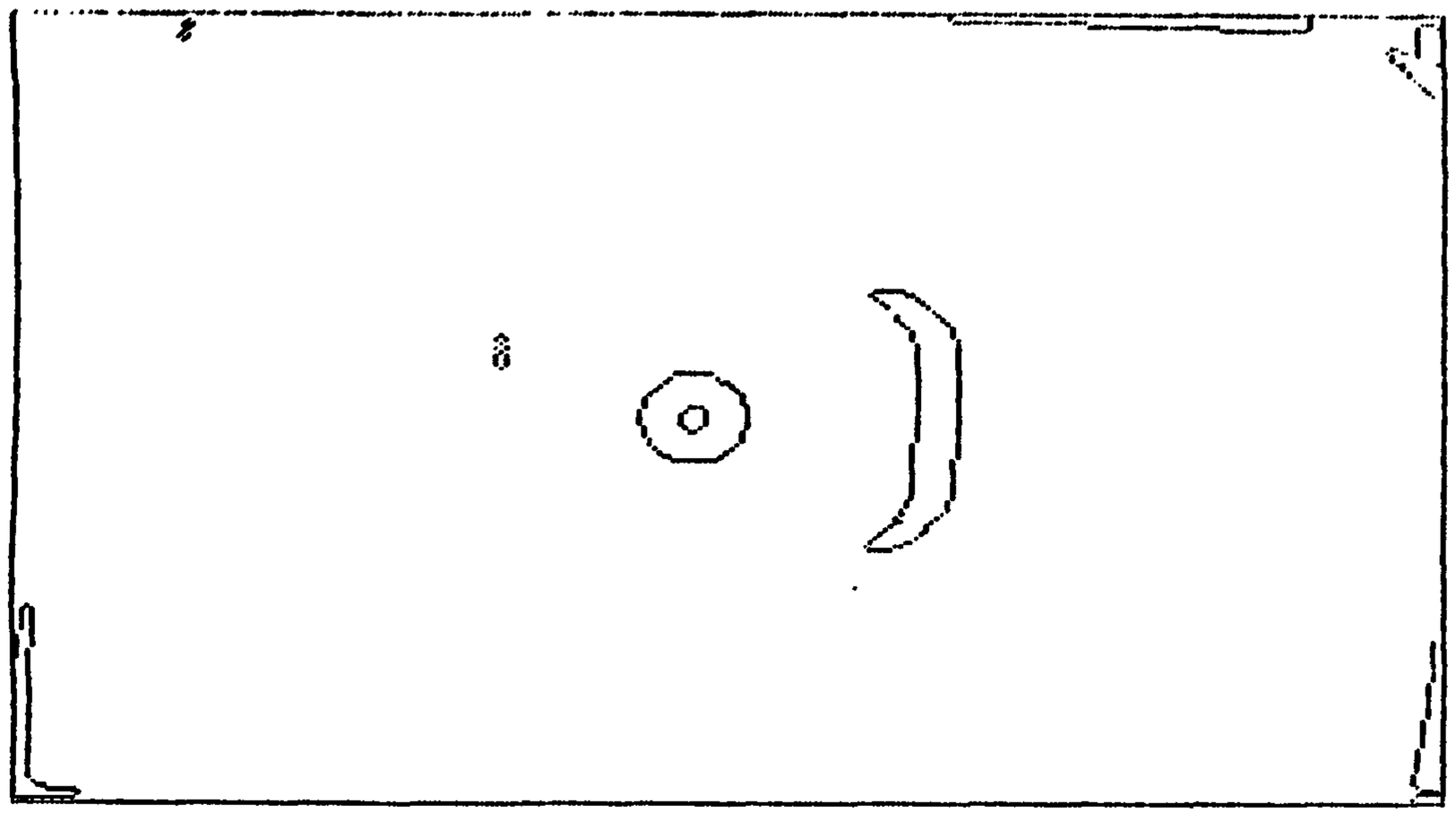


Figure 12.12a

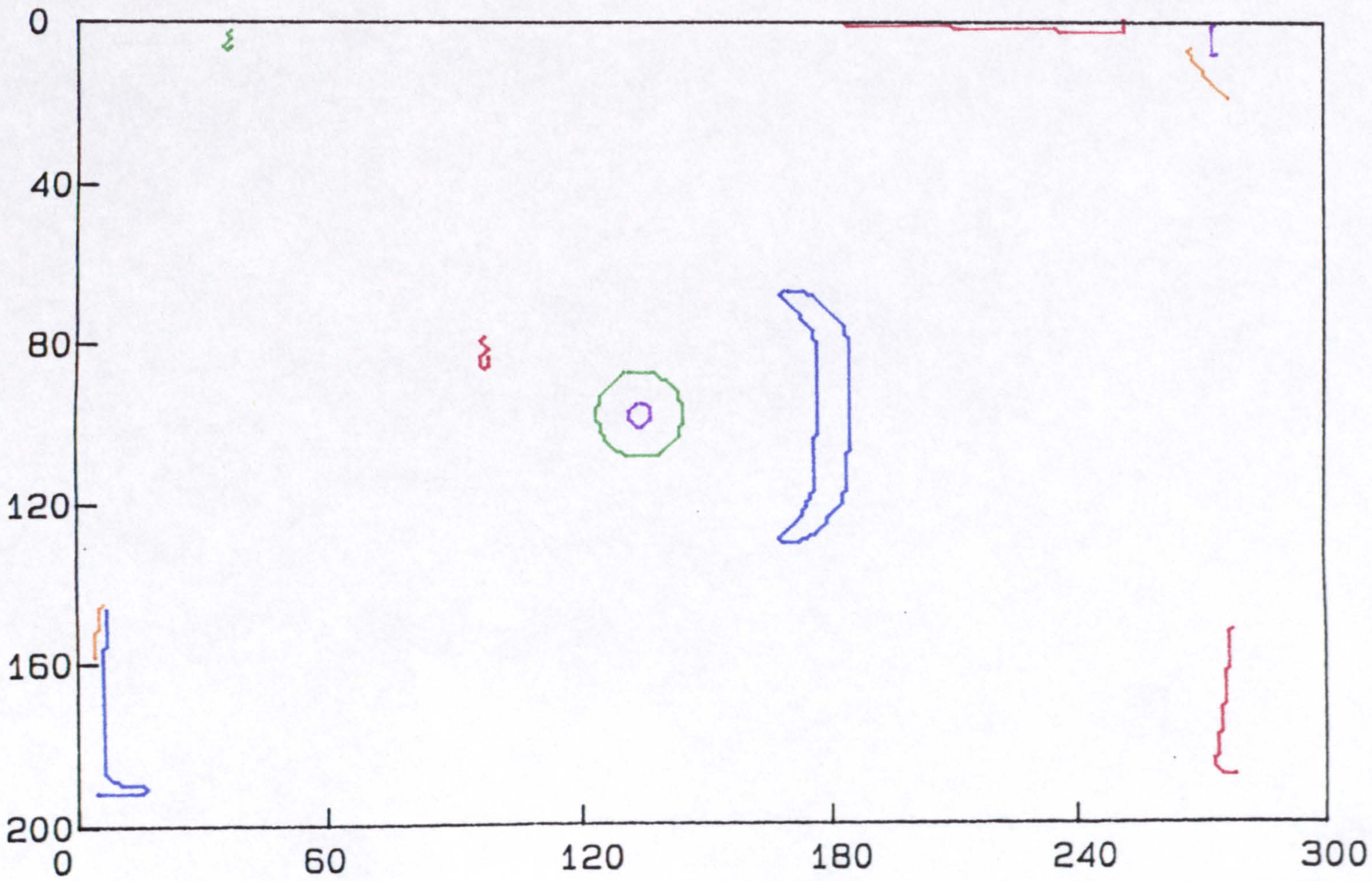
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0	0
10	167	68	1	176	80	13	168	127	60	138	151	501	45	15	47	59	62
14	0	0	0	0	0	0	0	0	0	59	69	436	10	0	0	0	0
17	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.12b



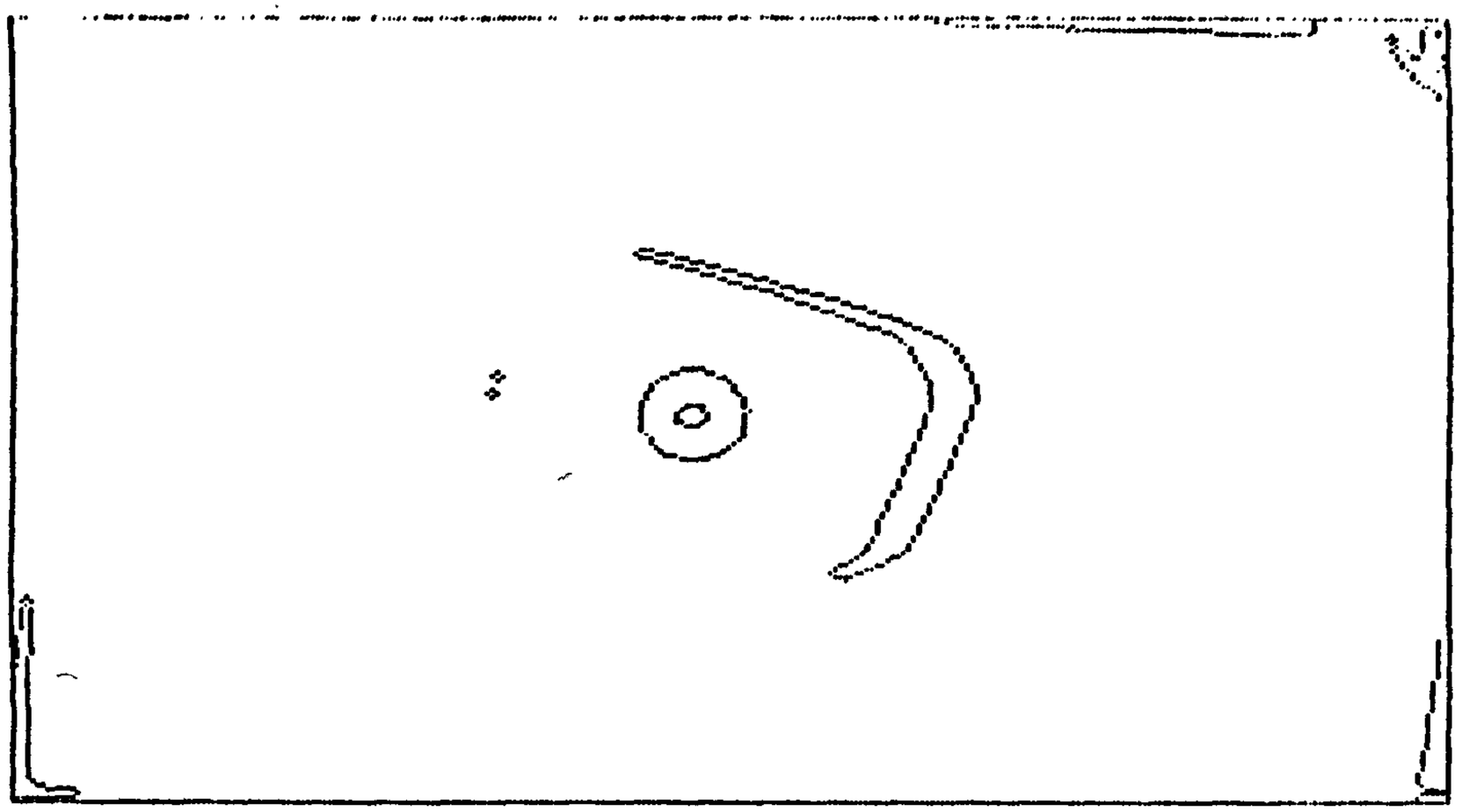


Figure 12.13a

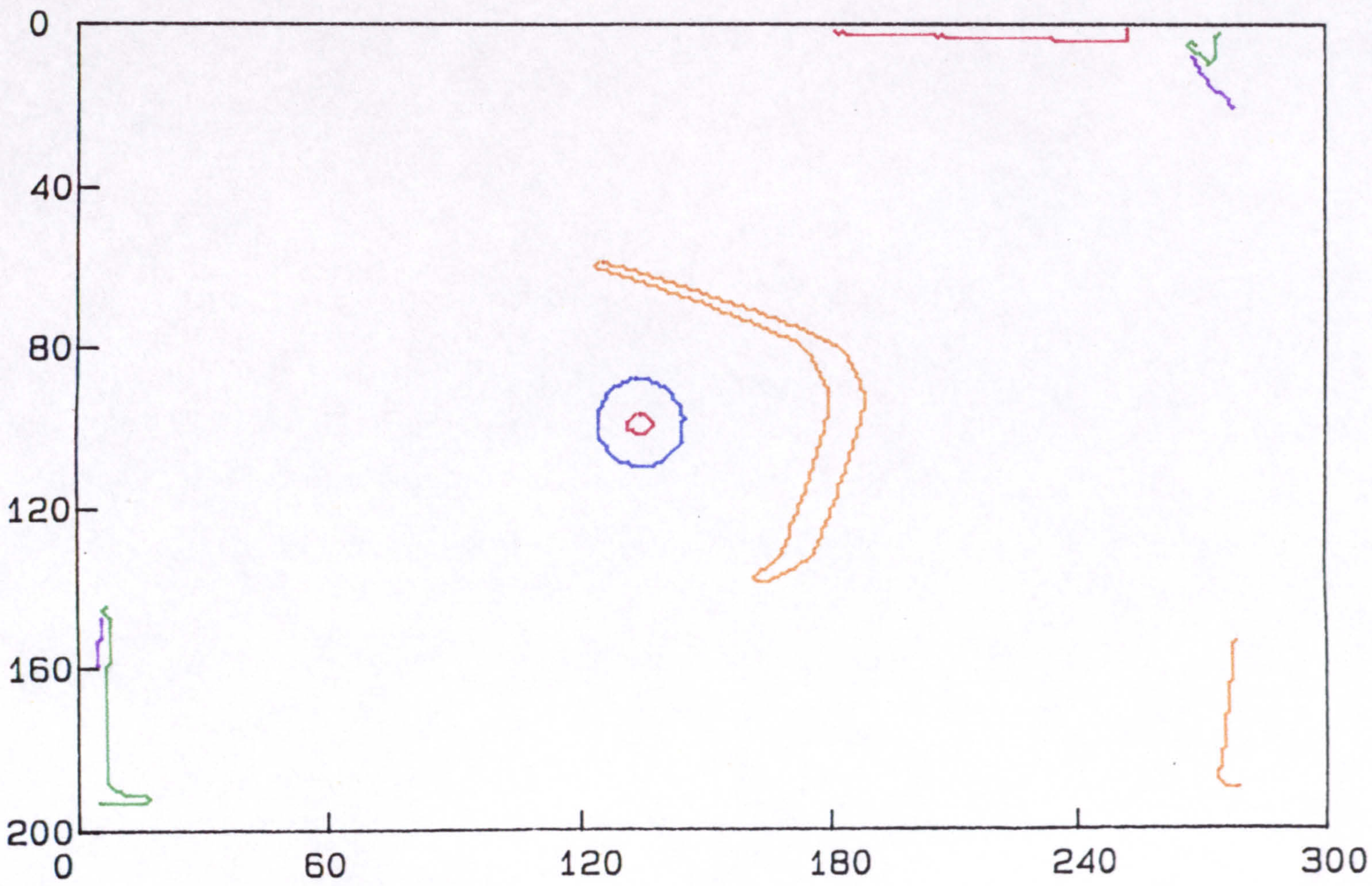
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0
7	121	58	1	178	89	62	161	134	108	235	270	644	113	64	48	85	112
8	0	0	0	0	0	0	0	0	0	60	70	439	11	0	0	0	0
13	0	0	0	0	0	0	0	0	0	73	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.13b



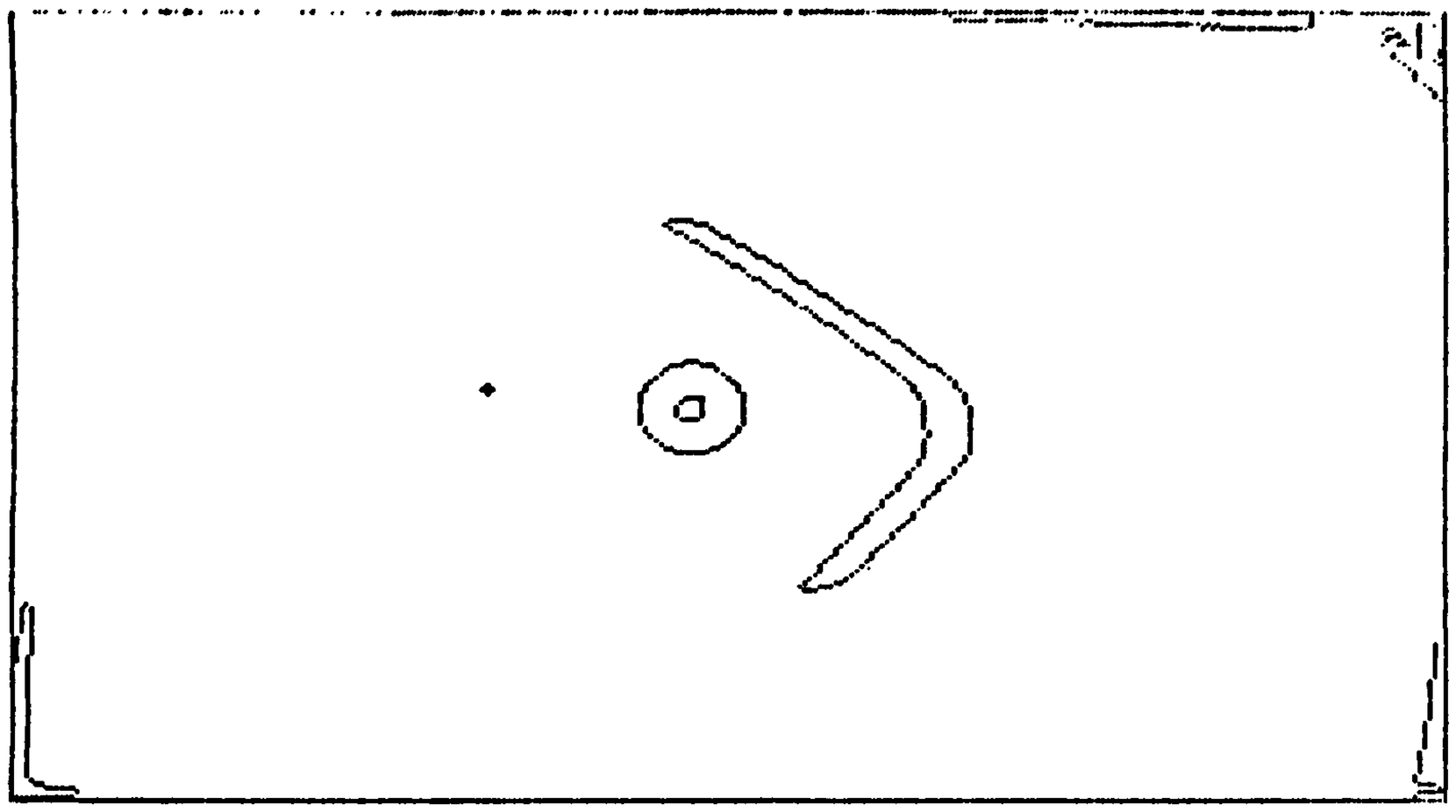


Figure 12.14a

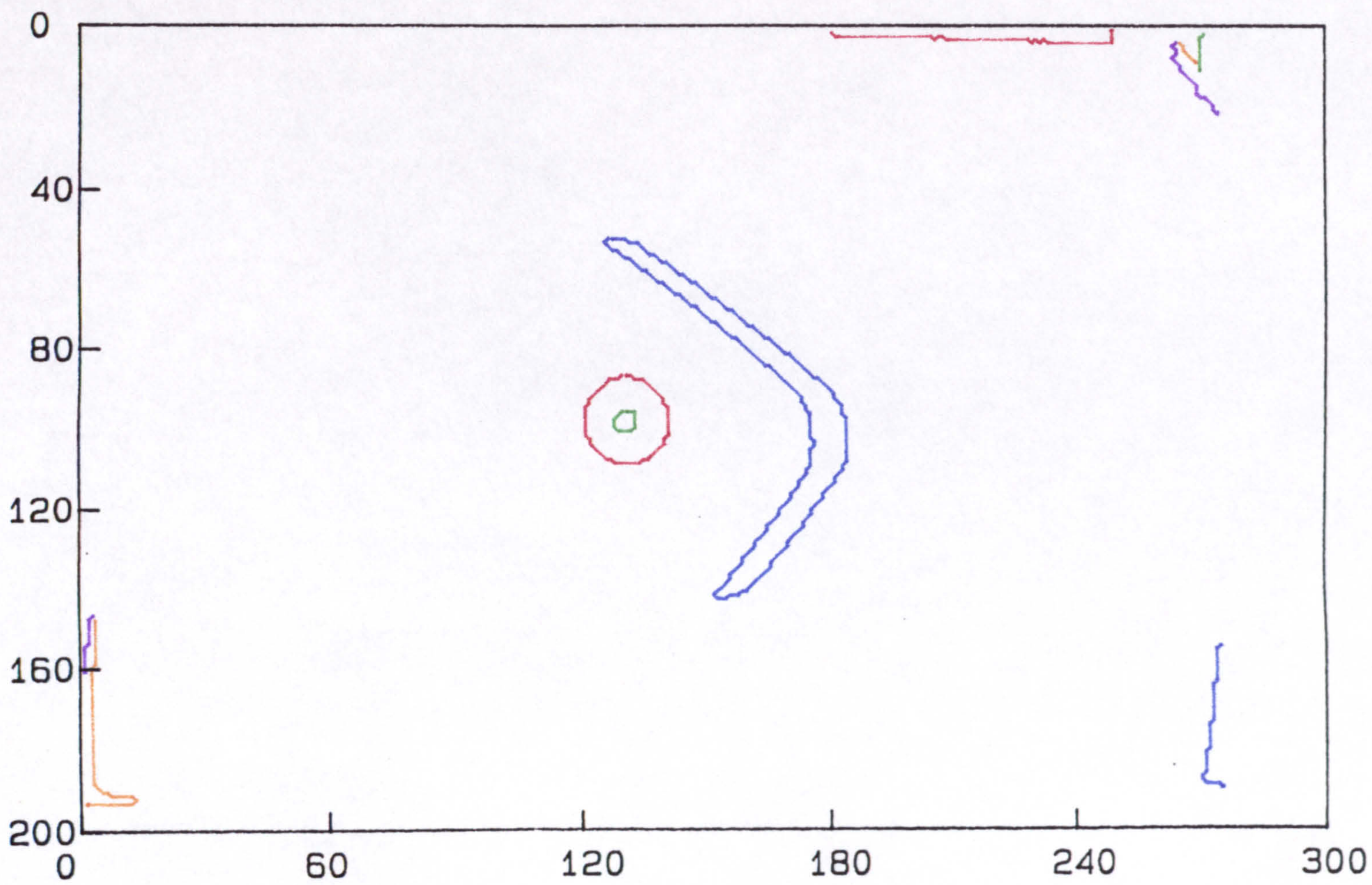
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0	0
8	127	52	1	177	95	53	155	138	96	210	262	719	95	65	48	90	113
9	0	0	0	0	0	0	0	0	0	58	68	430	10	0	0	0	0
13	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.14b



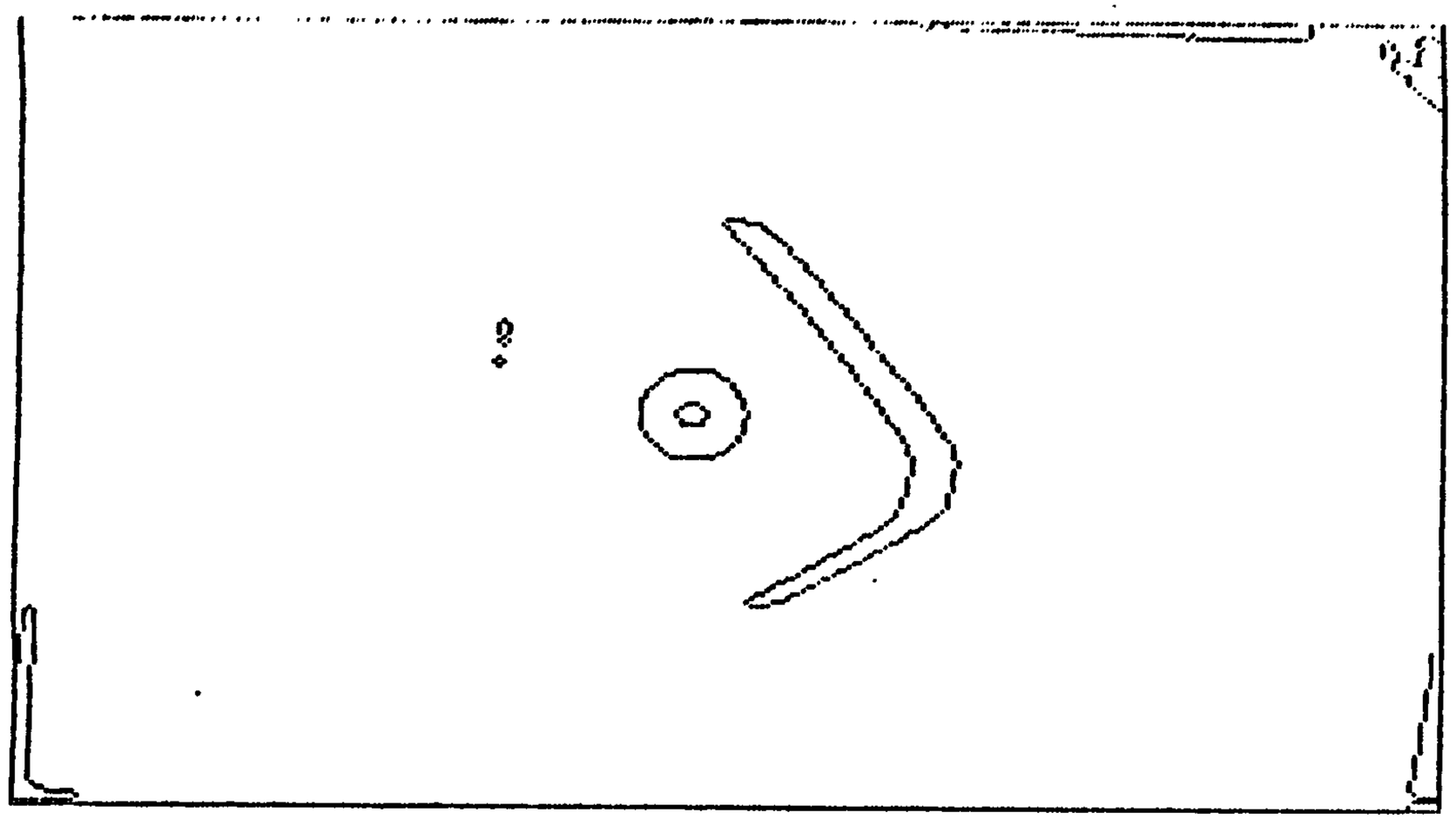


Figure 12.15a

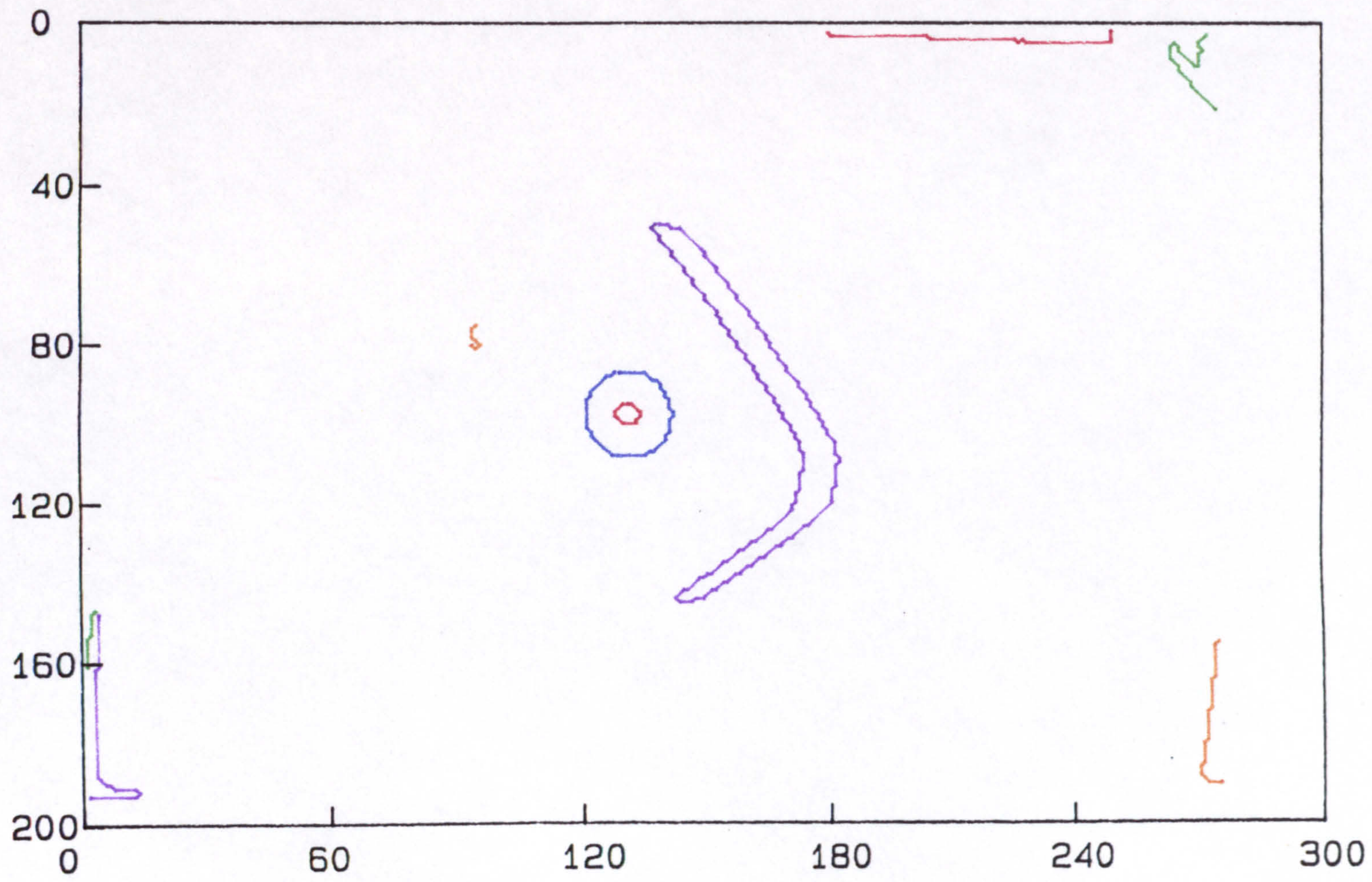
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	75	0	0	0	0	0	0	0
8	138	49	1	175	106	58	145	140	99	215	264	762	91	67	45	91	112
12	0	0	0	0	0	0	0	0	0	60	69	436	10	0	0	0	0
16	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.15b



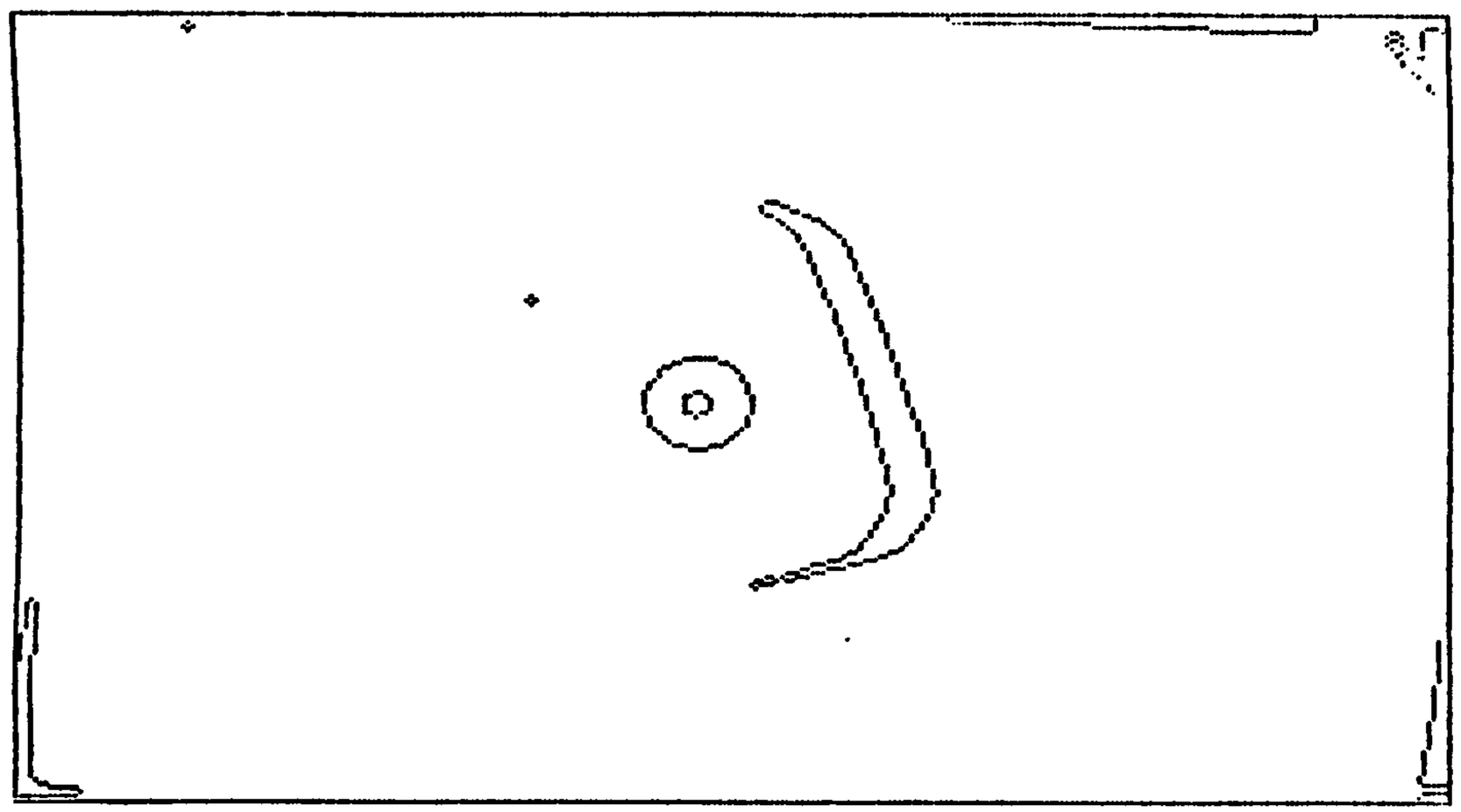


Figure 12.16a

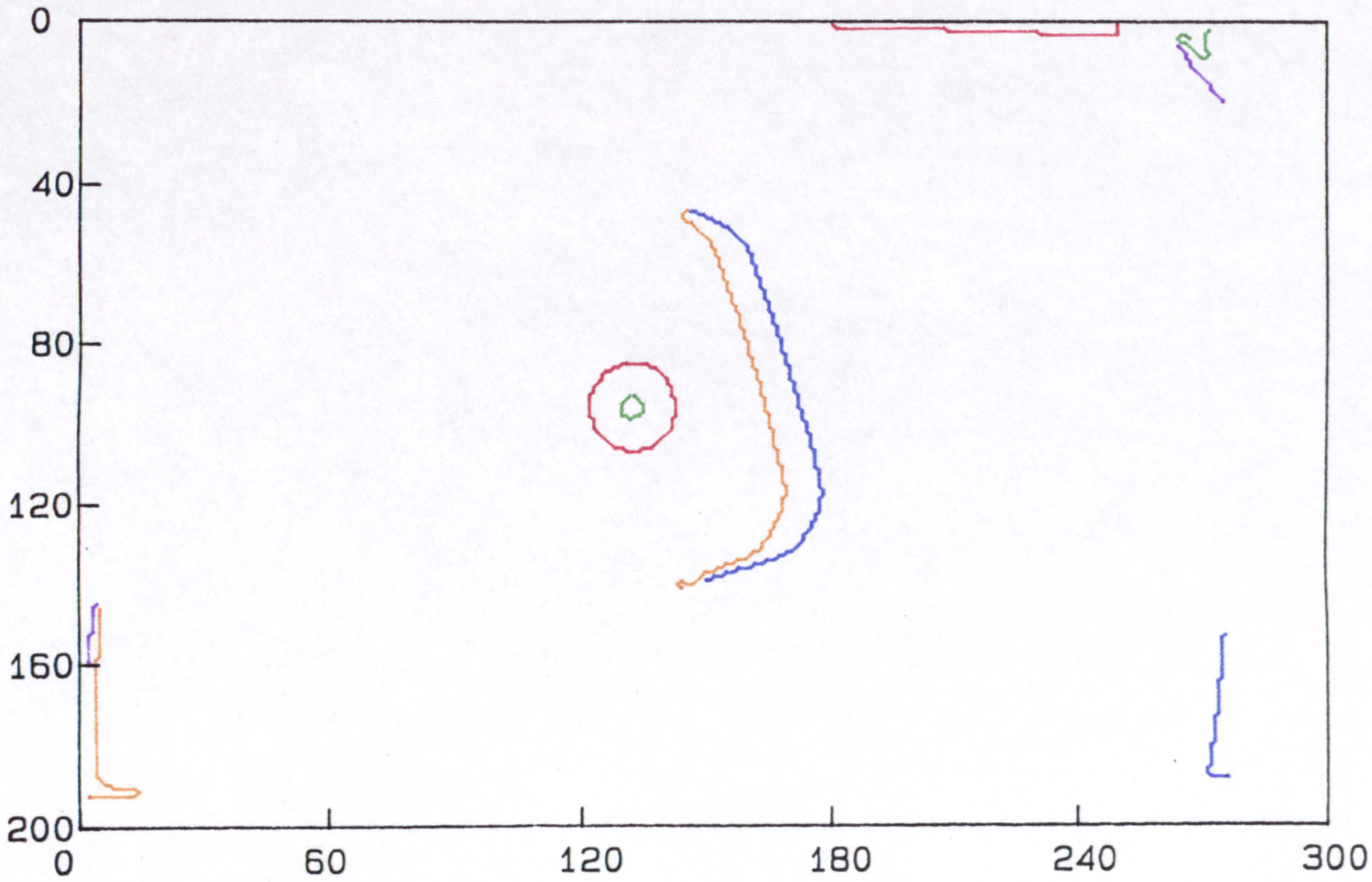
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
0	0	0	0	0	0	0	0	0	0	75	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	108	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	112	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	60	70	441	11	0	0	0	0
16	0	0	0	0	0	0	0	0	0	71	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.16b



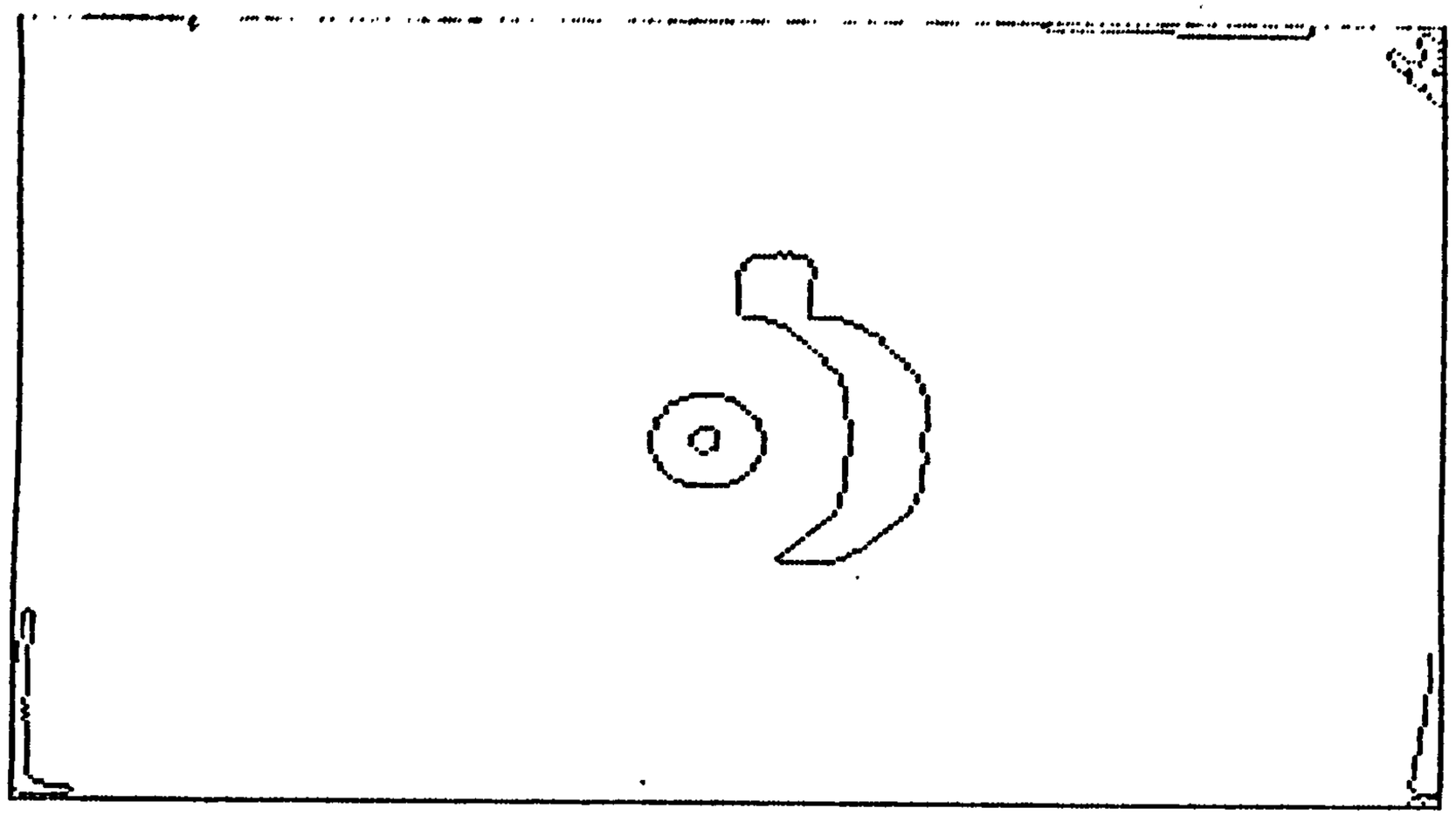


Figure 12.17a

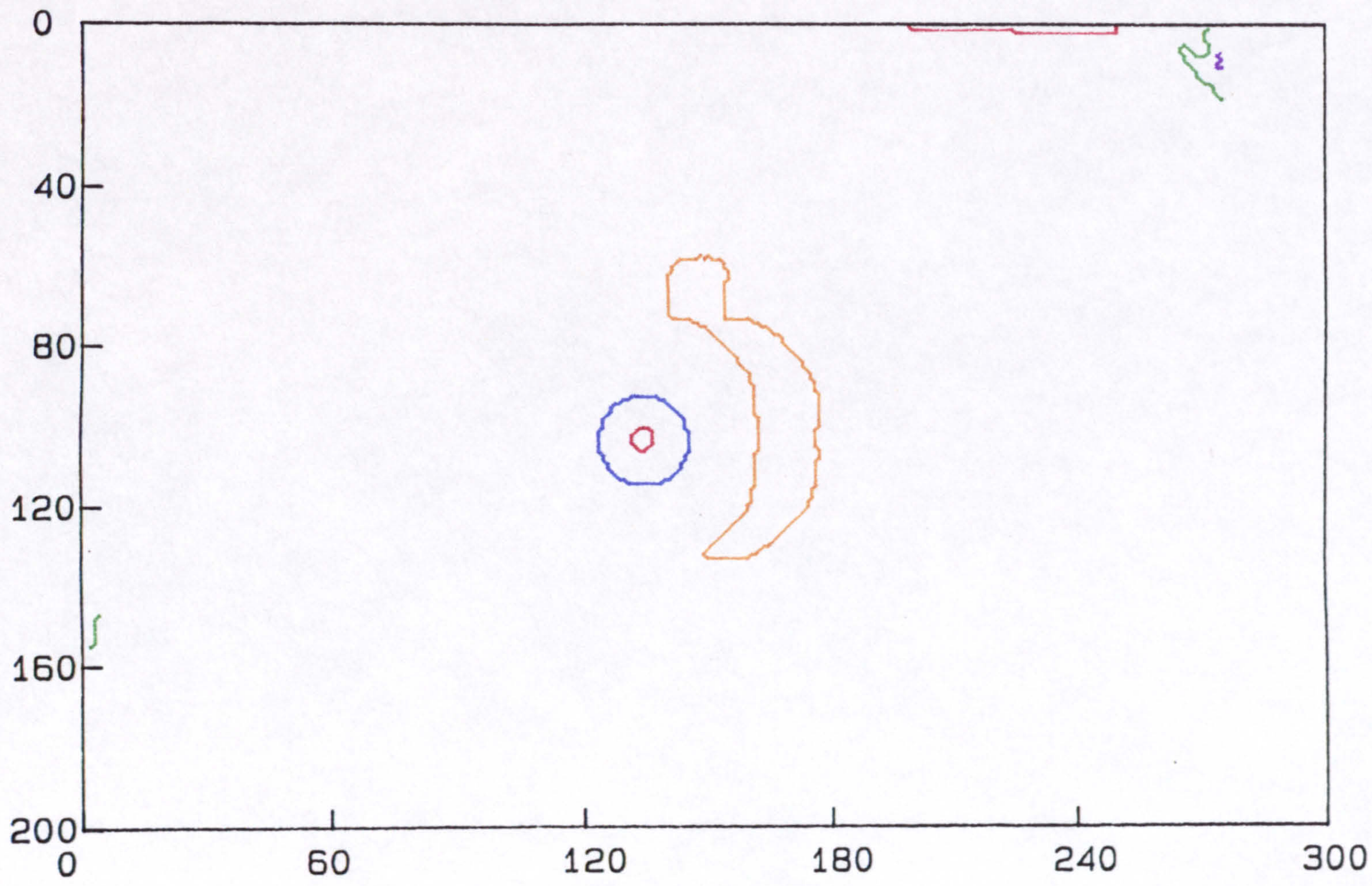
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	55	0.	0	0	0	0	0	0
9	141	72	19	163	98	51	151	130	83	187	216	1154	40	34	34	58	68
10	0	0	0	0	0	0	0	0	0	63	73	478	11	0	0	0	0
13	0	0	0	0	0	0	0	0	0	109	119	380	37	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.17b



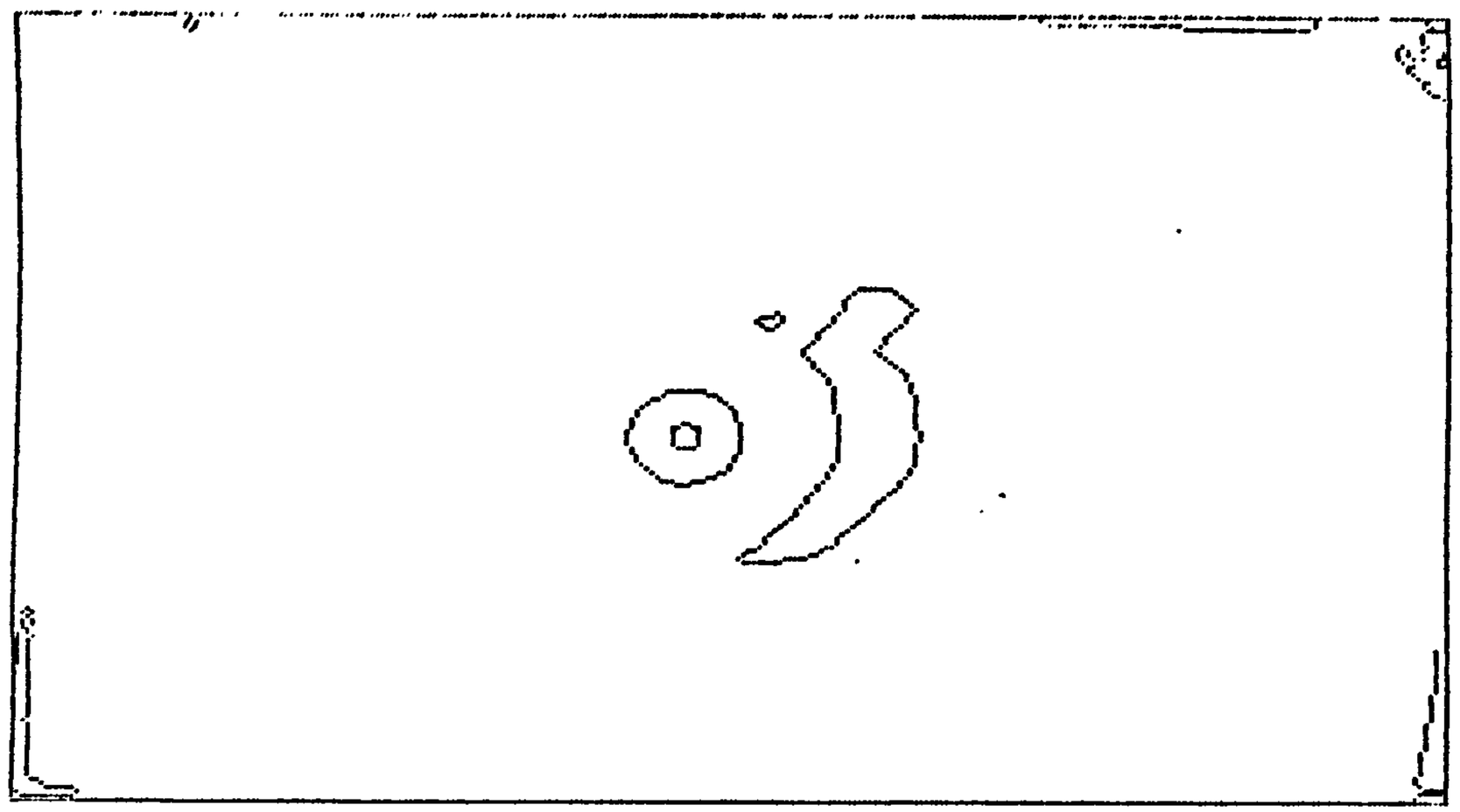


Figure 12.18a

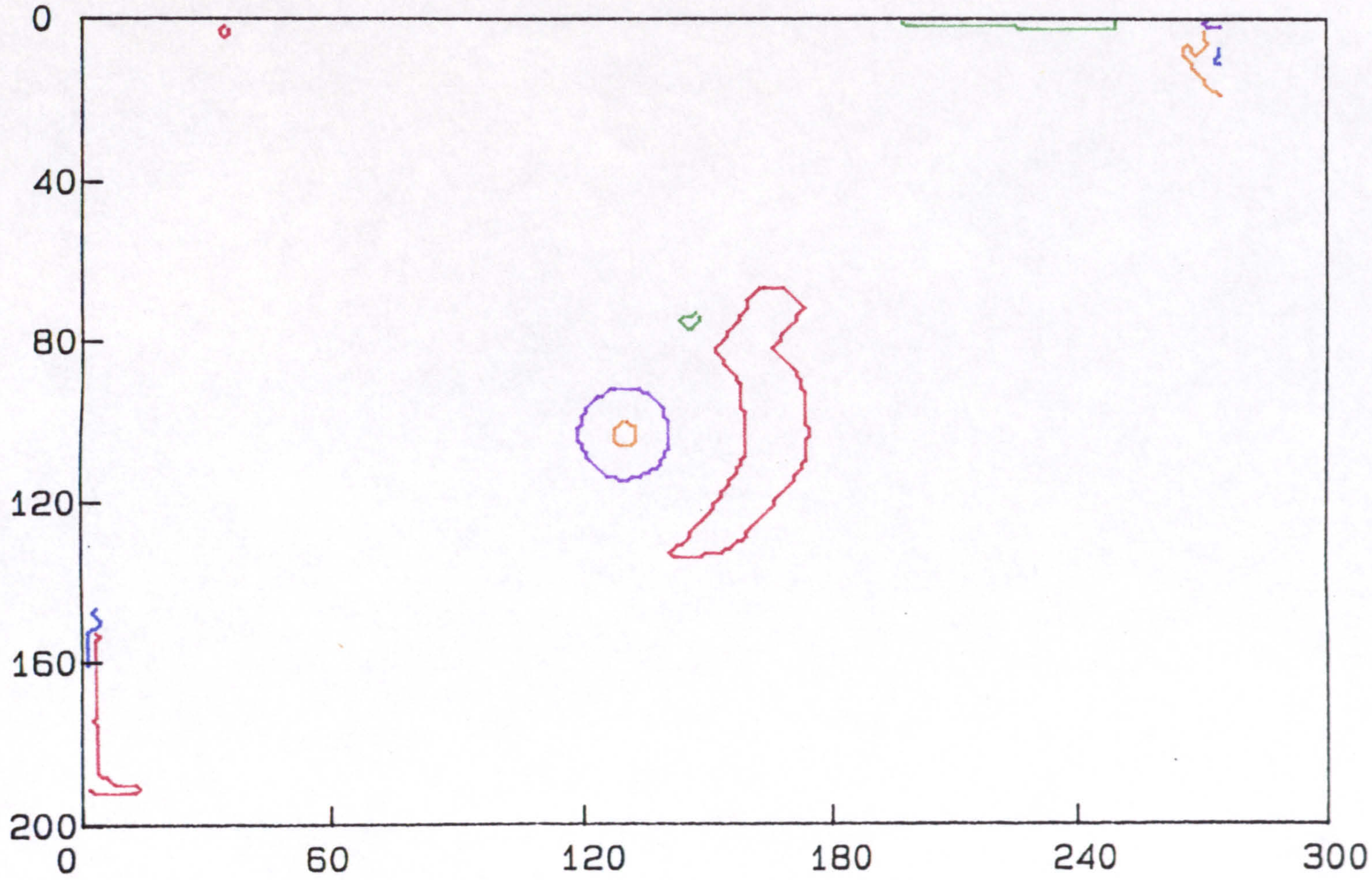
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0
10	153	82	16	160	97	31	143	130	65	154	186	1030	33	16	37	49	53
12	0	0	0	0	0	0	0	0	0	63	74	480	11	0	0	0	0
18	1	173	22	10	189	43	1	191	56	97	59	-26	-133	18	9	18	27

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.18b



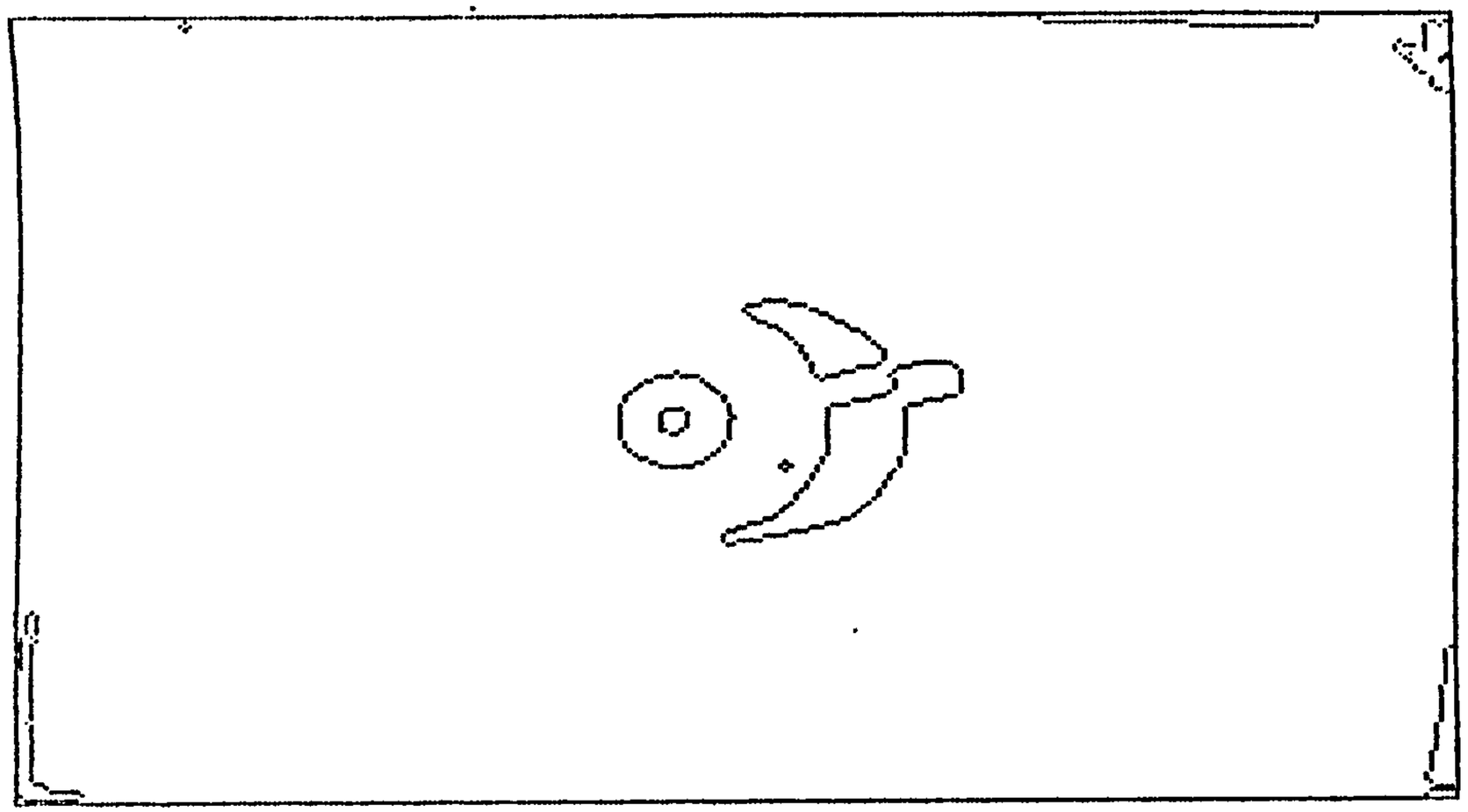


Figure 12.19a

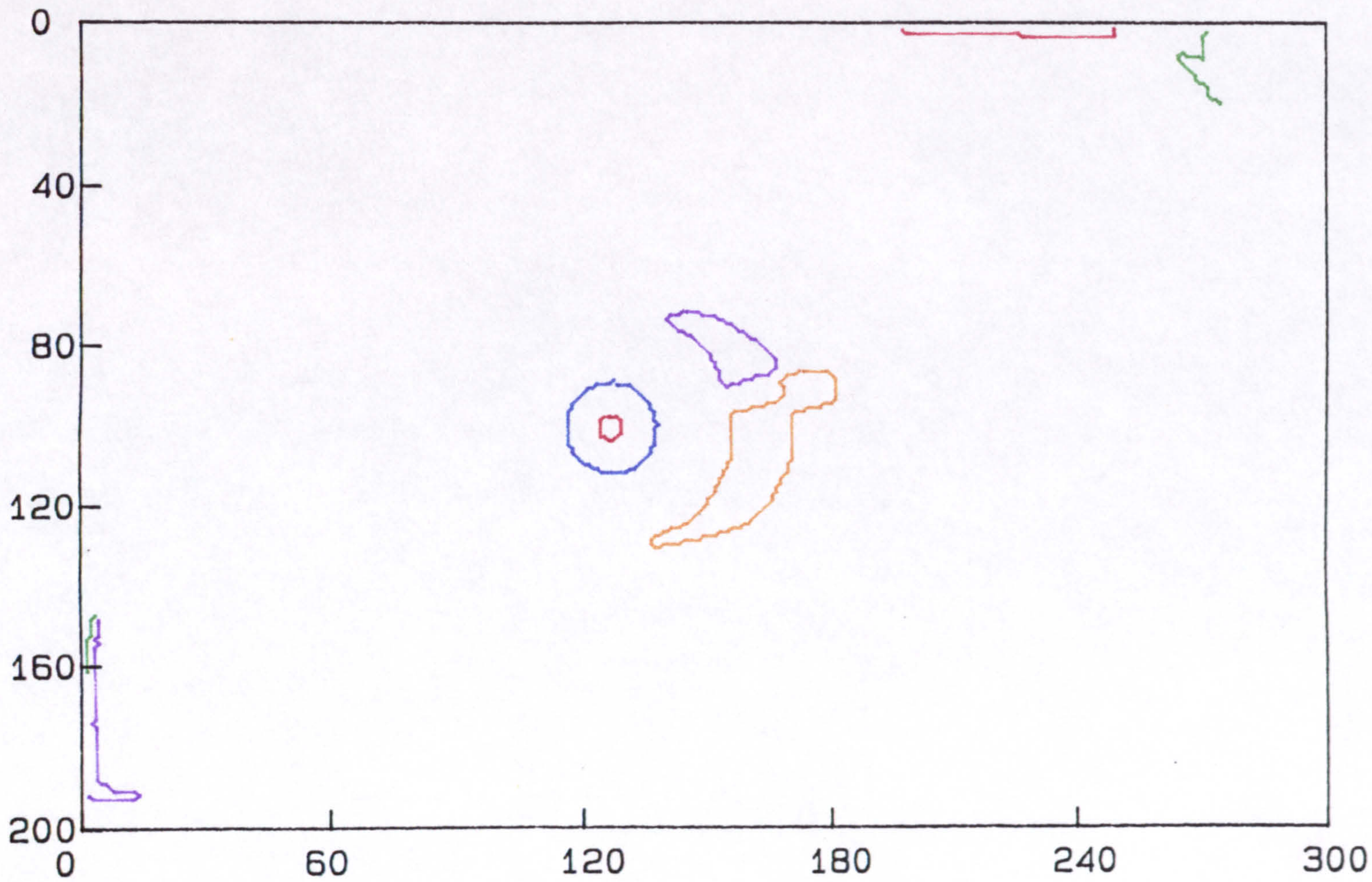
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	j	pid	midpoint i	j	pid	maxpoint i	j	pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	60	73	307	17	0	0	0	0
8	0	0	0	0	0	0	0	0	0	134	153	701	33	0	0	0	0
9	0	0	0	0	0	0	0	0	0	62	73	470	11	0	0	0	0
13	0	0	0	0	0	0	0	0	0	101	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.19b



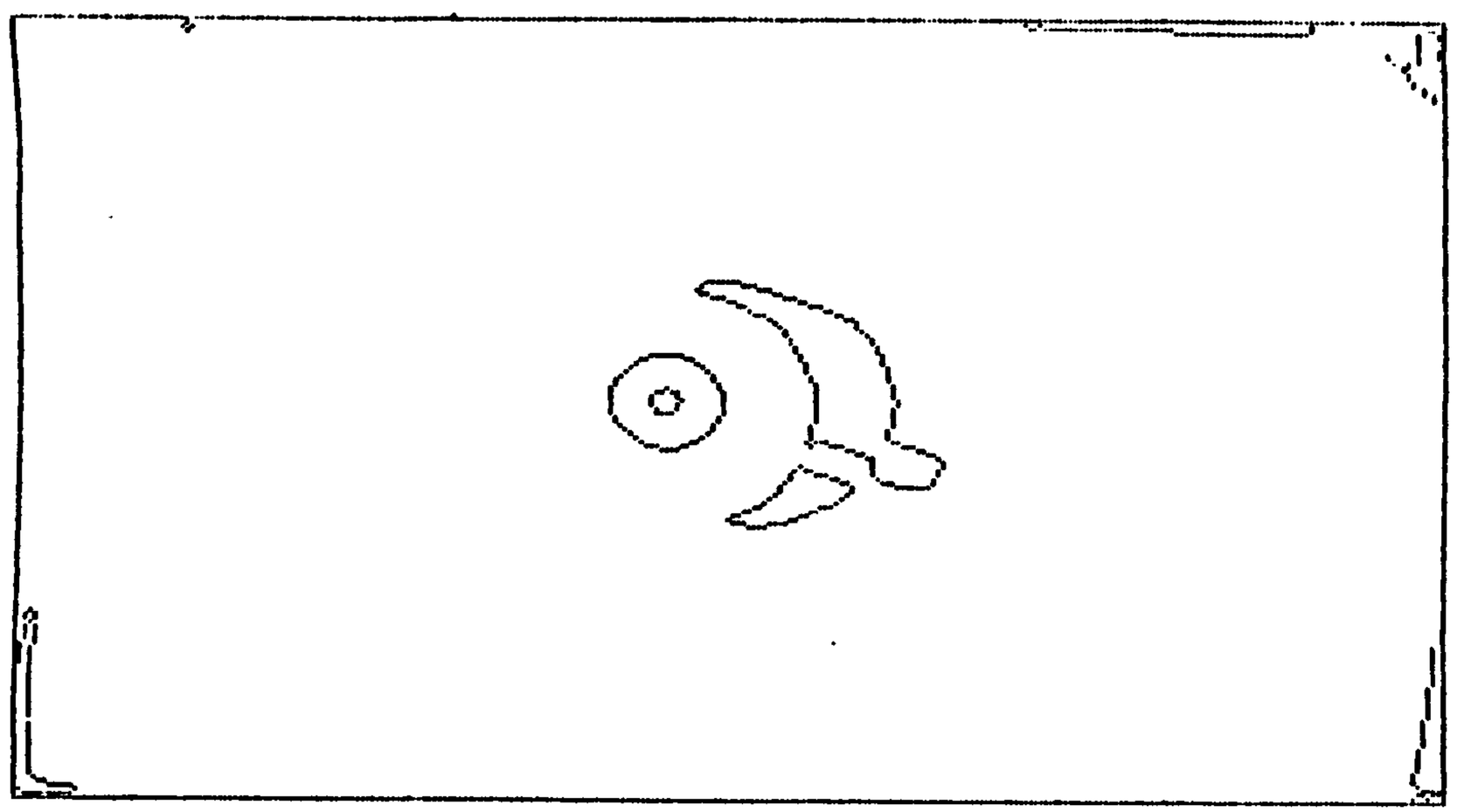


Figure 12.20a

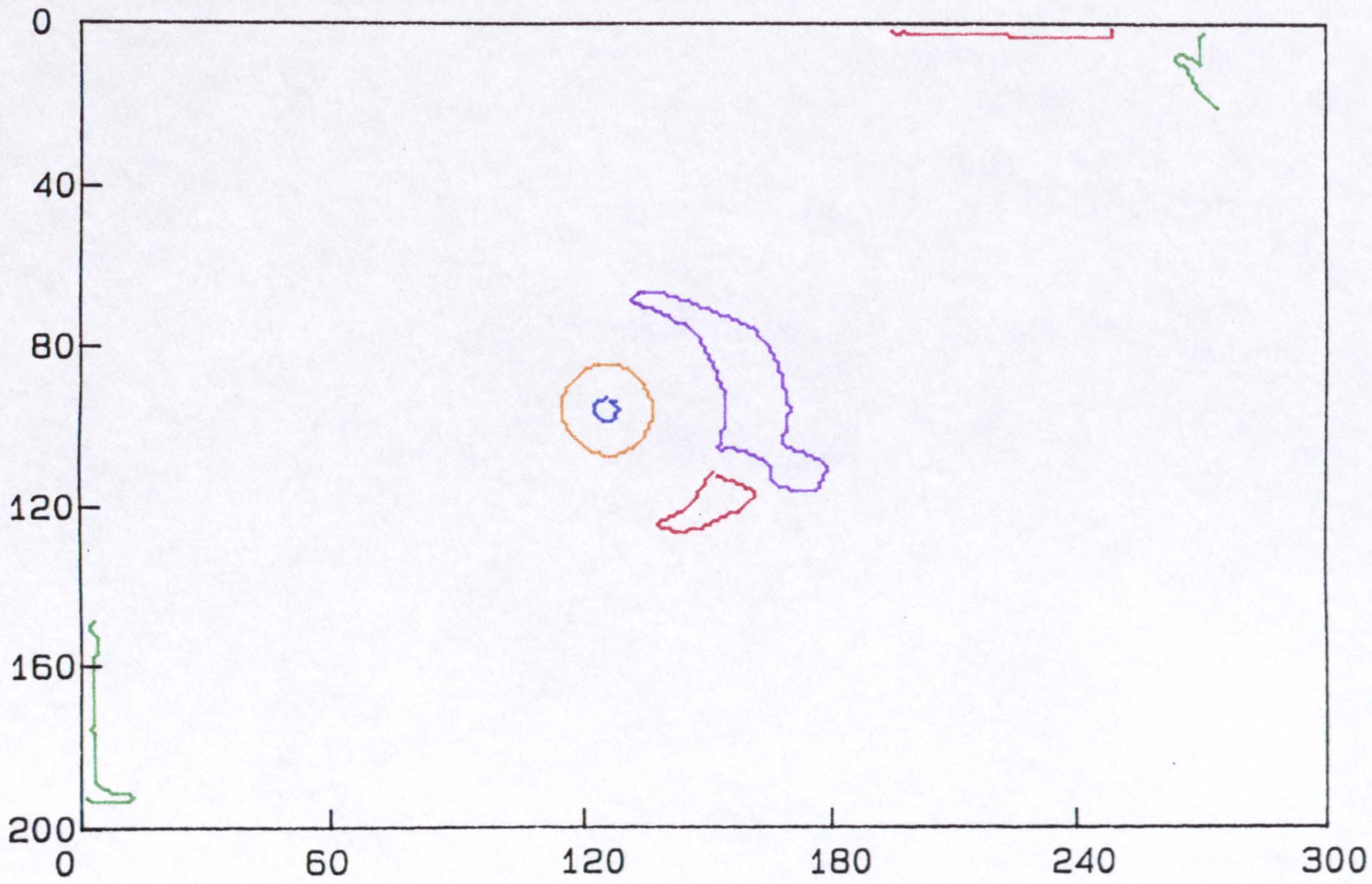
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	0	0
9	133	67	2	156	90	32	156	99	41	147	164	718	37	32	9	39	41
10	0	0	0	0	0	0	0	0	0	61	72	468	11	0	0	0	0
14	0	0	0	0	0	0	0	0	0	51	61	270	13	0	0	0	0
15	1	148	2	9	189	47	1	191	59	106	63	-25	-158	41	8	43	49

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.20b



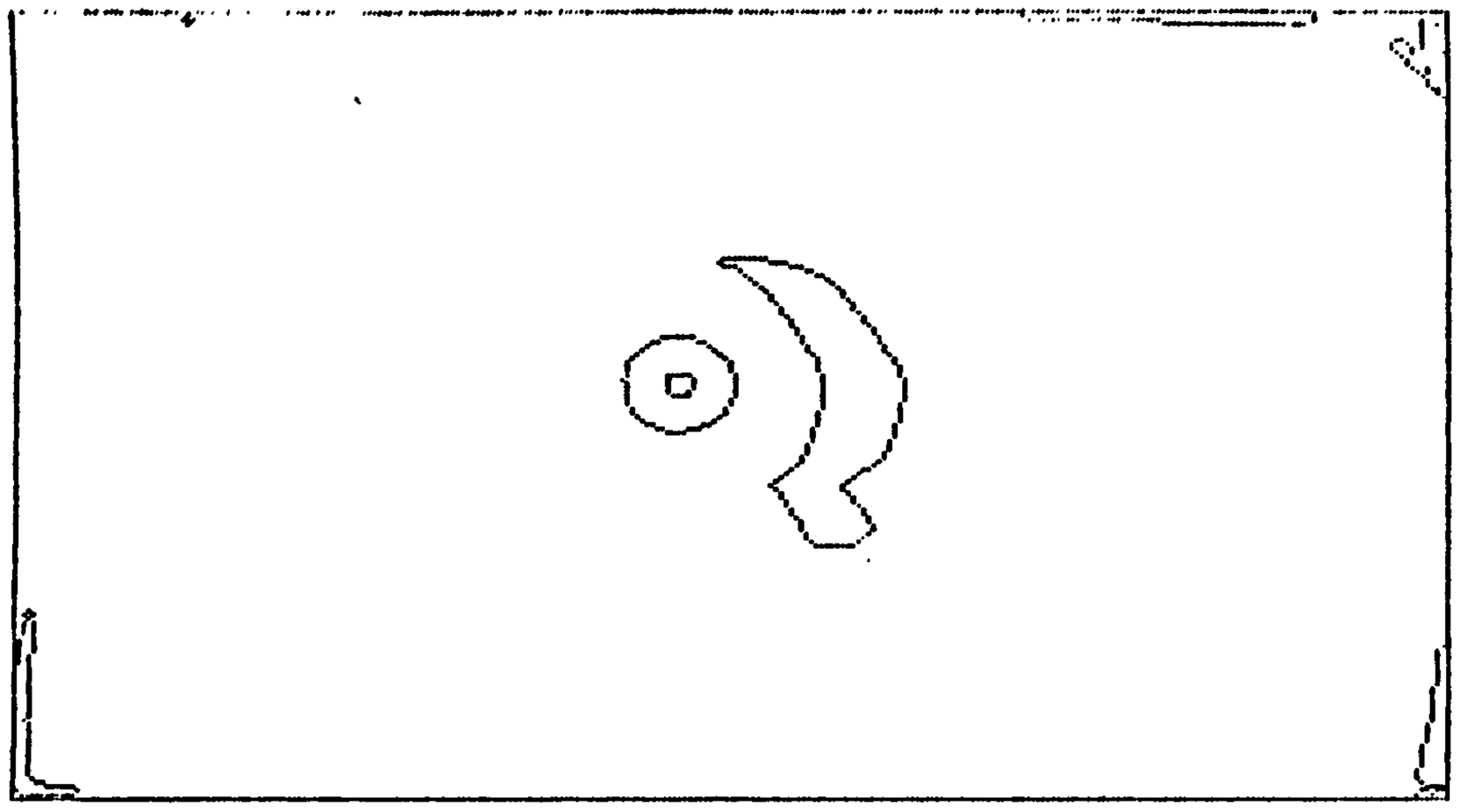


Figure 12.21a

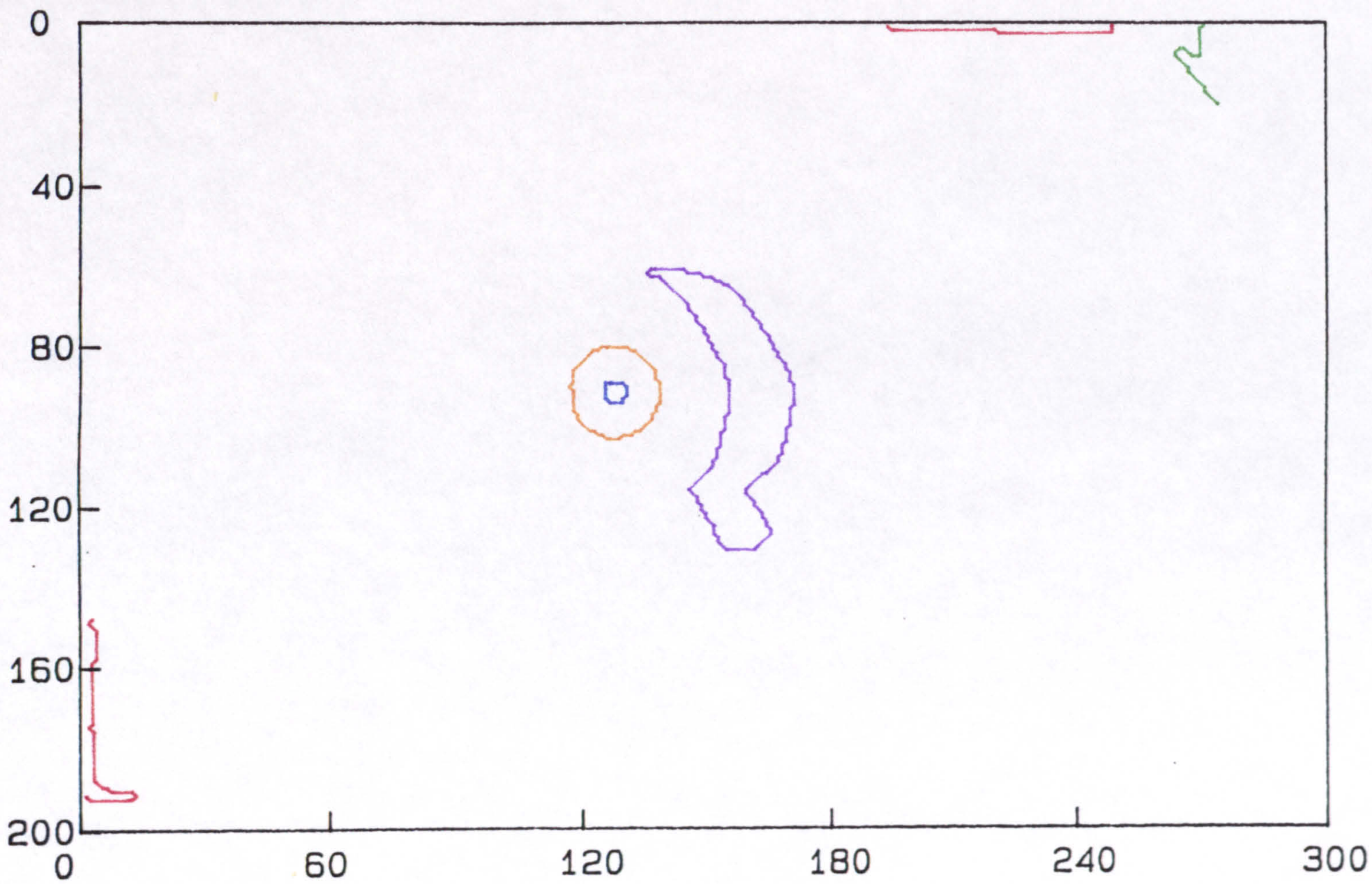
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0
8	137	61	1	157	88	30	149	113	55	165	193	1013	36	33	26	53	59
9	0	0	0	0	0	0	0	0	0	62	73	463	11	0	0	0	0
11	1	147	1	10	189	48	1	191	61	108	65	-31	-136	42	9	44	51

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.21b



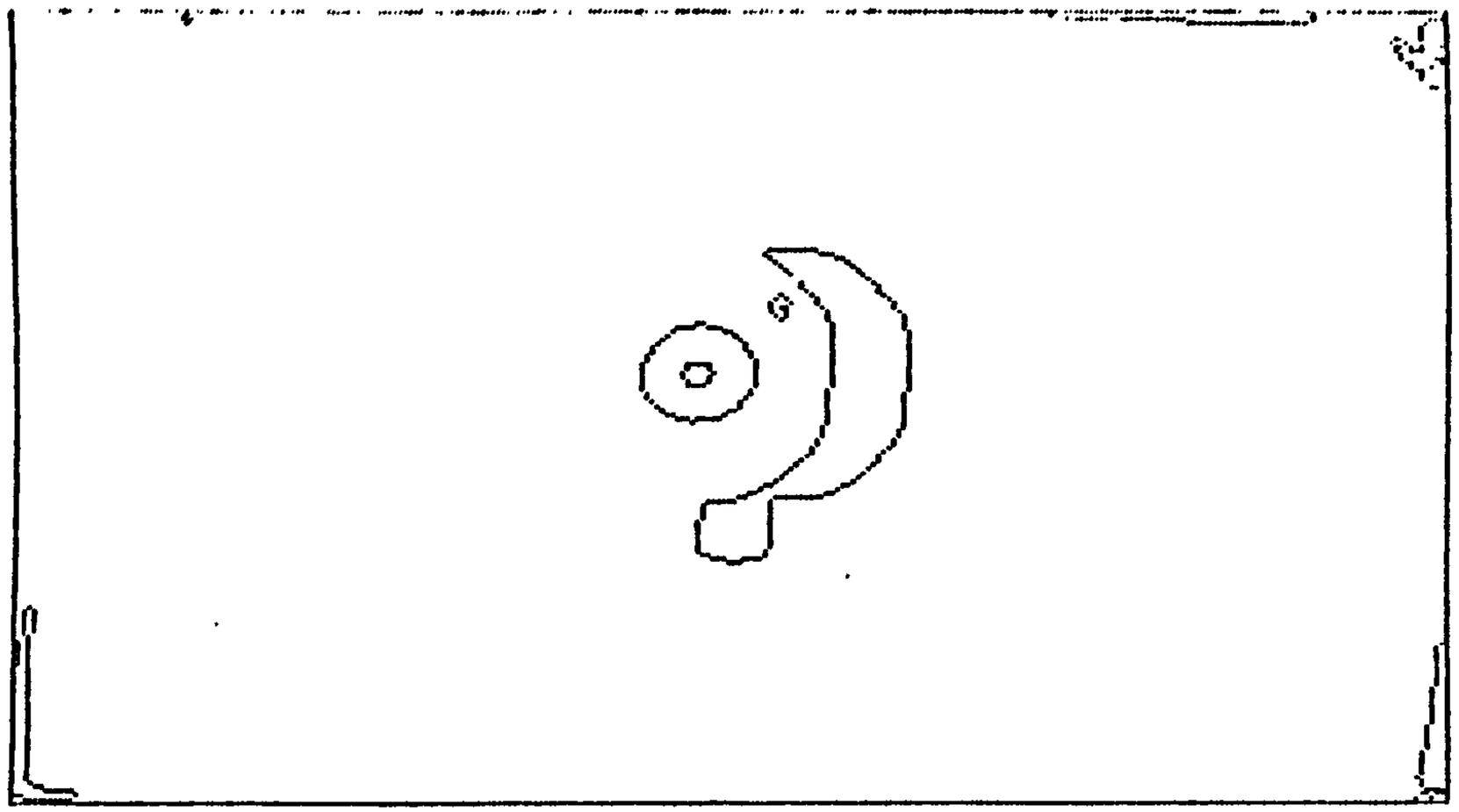


Figure 12.22a

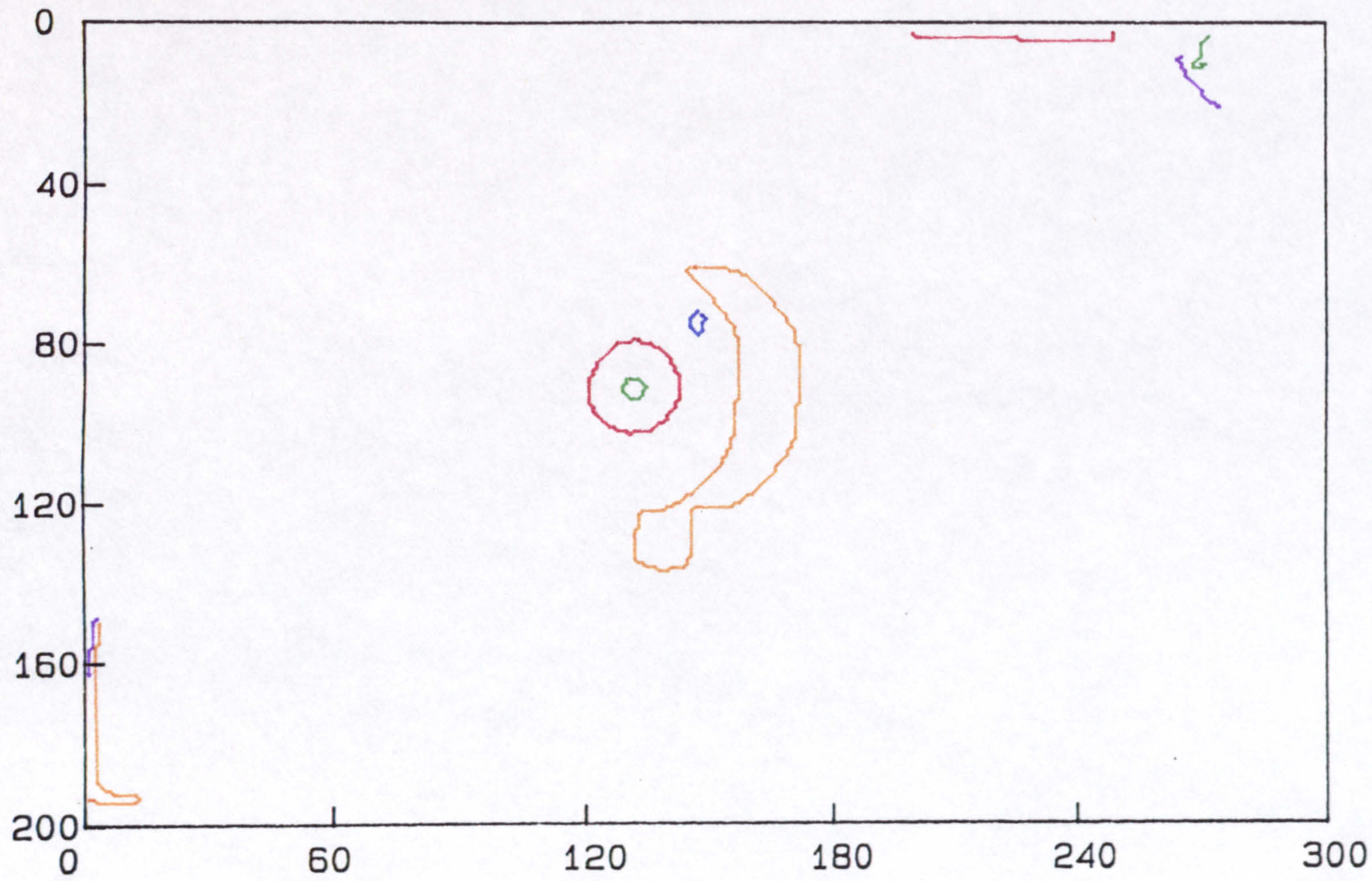
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0
11	146	59	1	159	76	18	134	123	73	191	215	1071	43	21	53	65	74
14	0	0	0	0	0	0	0	0	0	63	74	469	11	0	0	0	0
19	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.22b



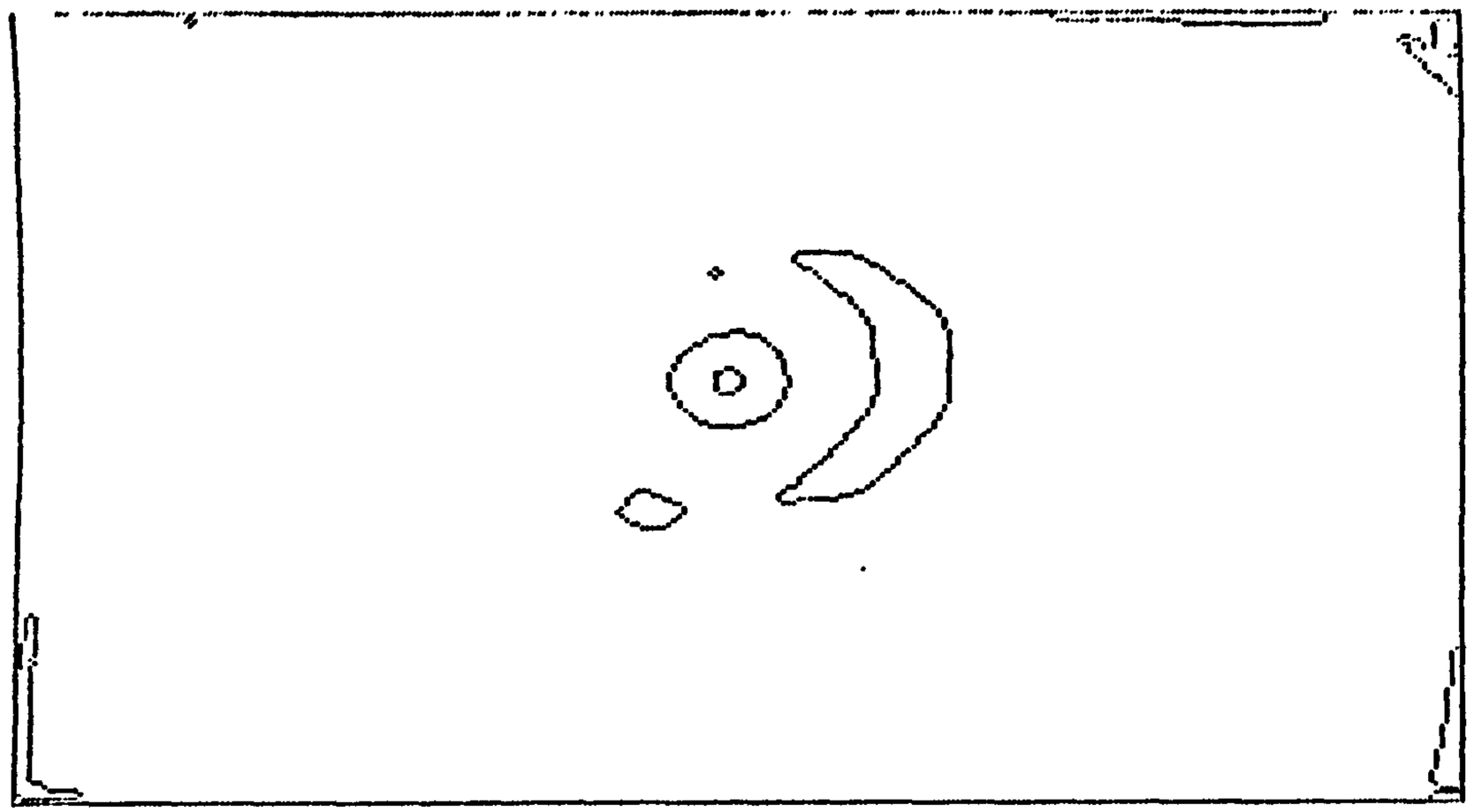


Figure 12.23a

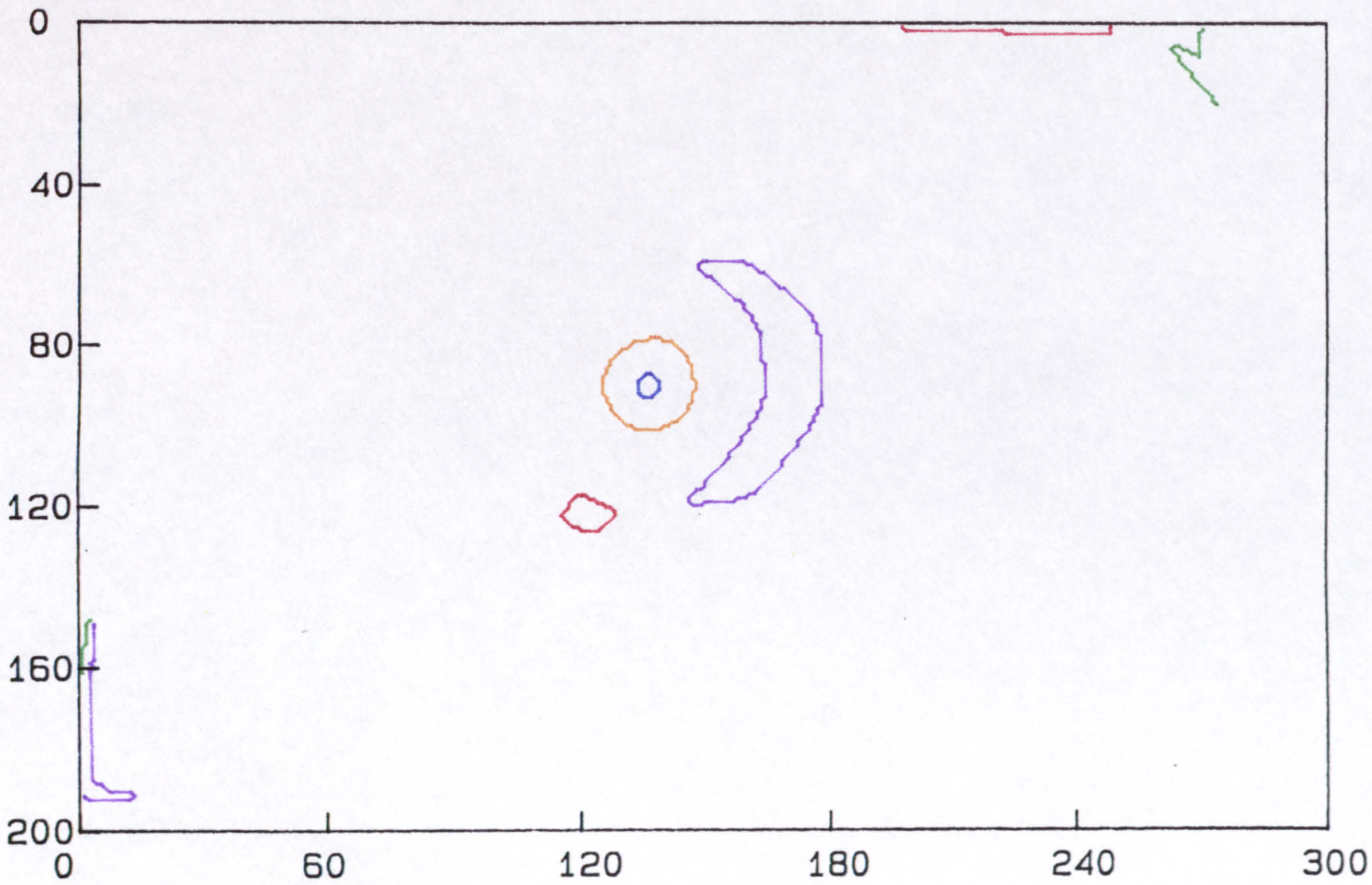
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0
9	150	60	2	166	84	28	149	115	60	150	173	865	34	28	35	55	63
11	0	0	0	0	0	0	0	0	0	63	74	473	11	0	0	0	0
16	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.23b



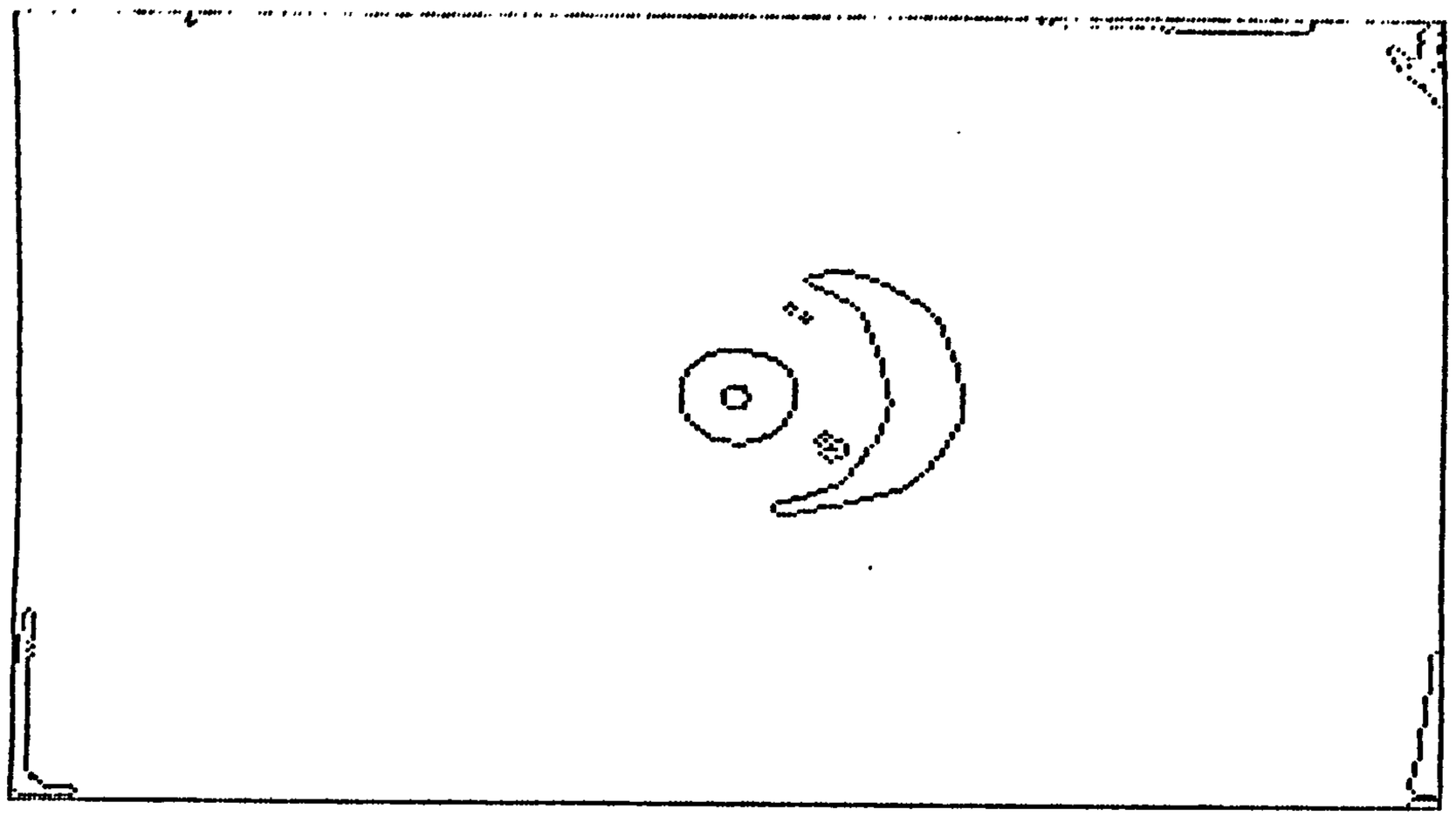


Figure 12.24a

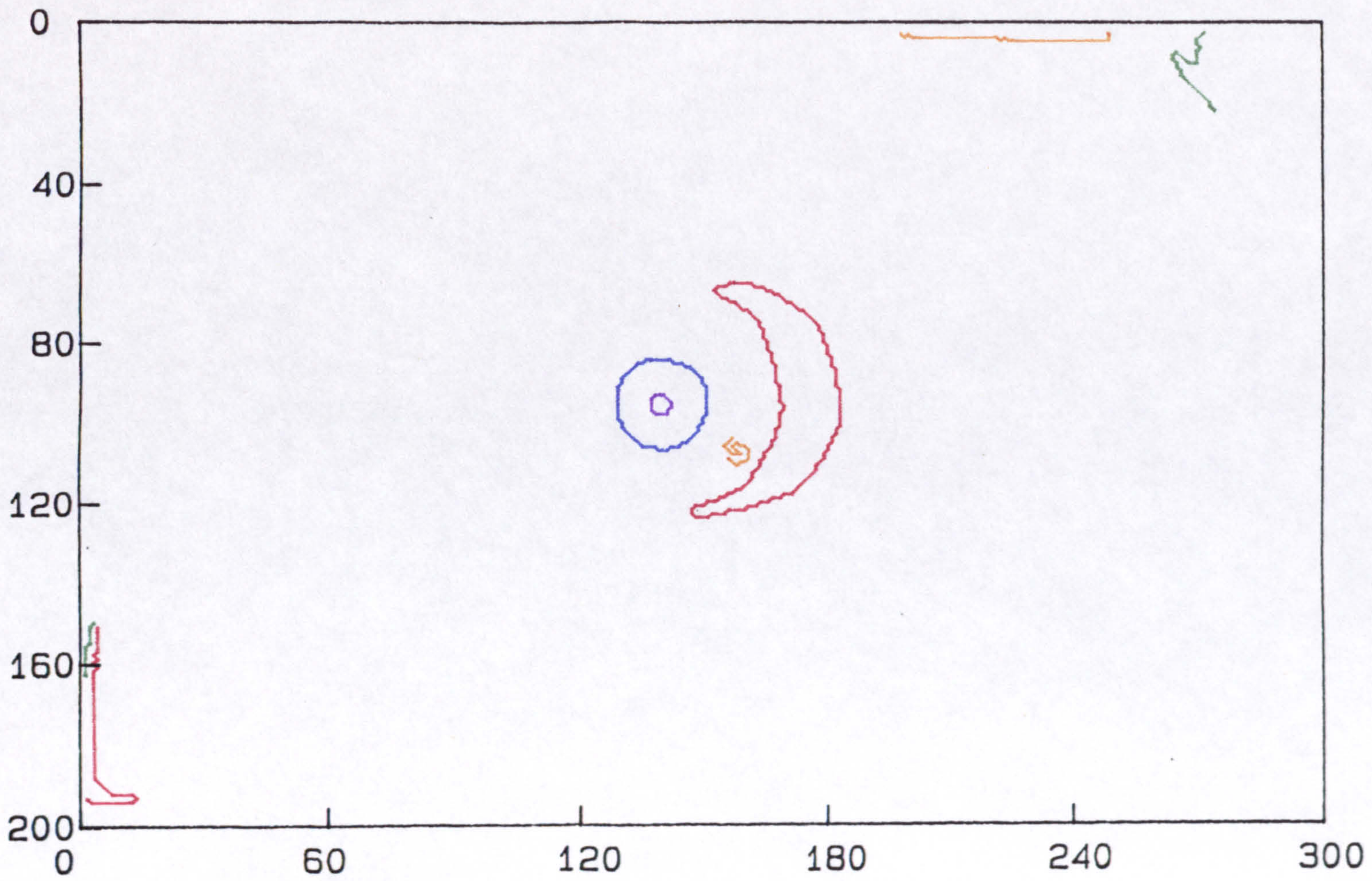
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	55	0.	0	0	0	0	0	0
8	154	64	4	170	88	31	150	118	67	159	186	889	38	28	36	54	64
11	0	0	0	0	0	0	0	0	0	64	74	484	11	0	0	0	0
16	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.24b



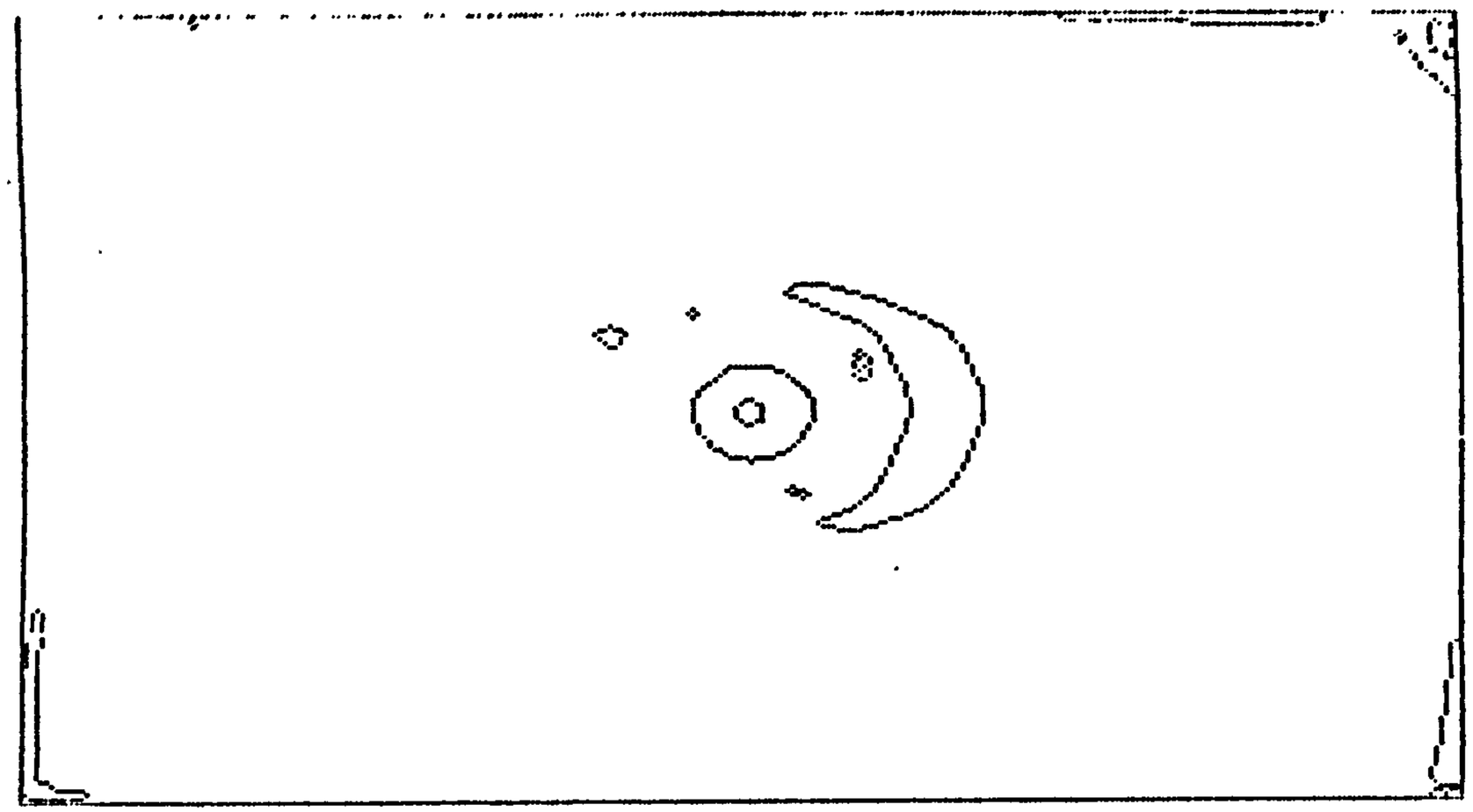


Figure 12.25a

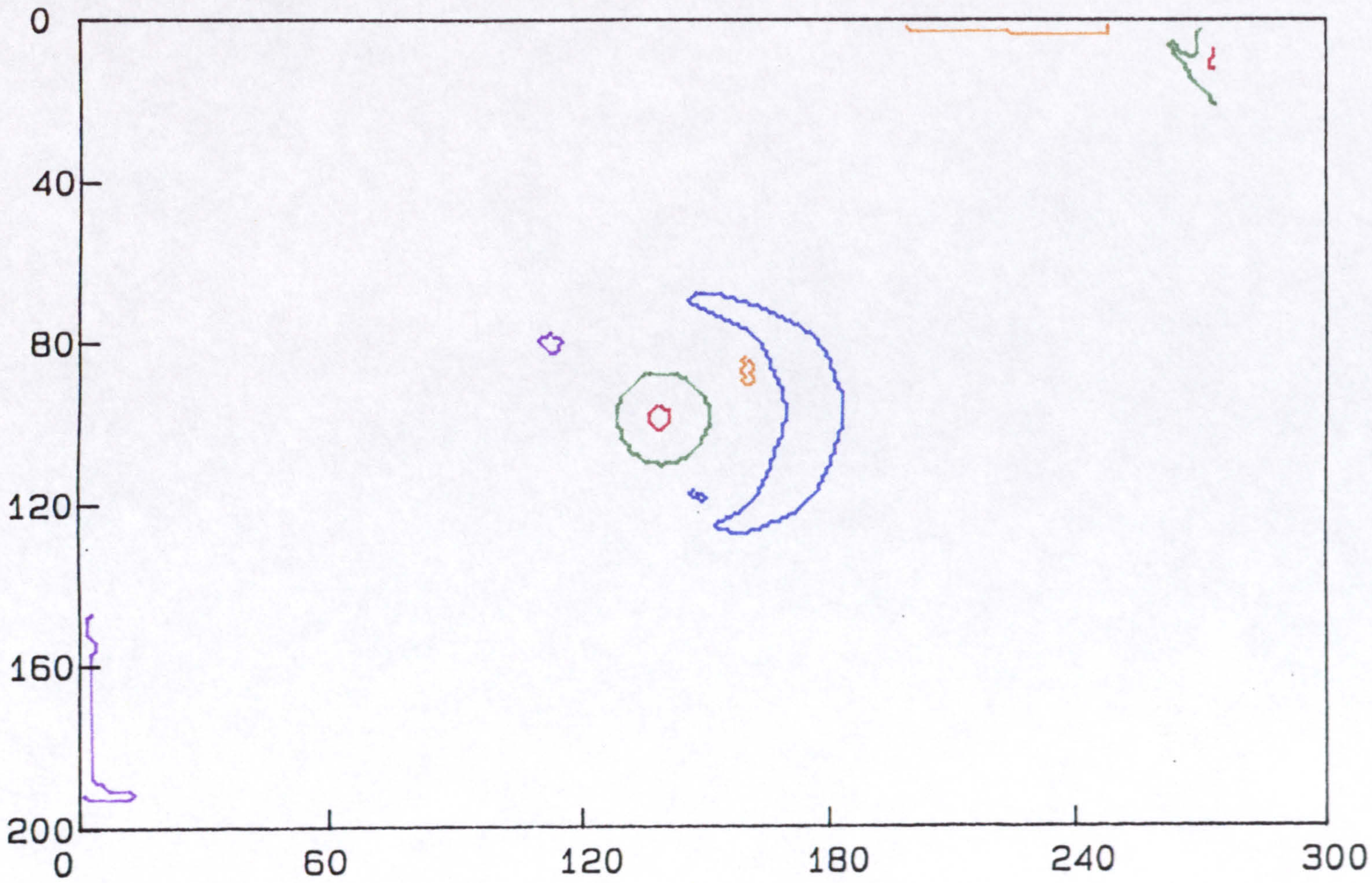
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0
8	148	68	2	172	94	35	156	123	66	161	183	840	39	35	33	55	68
12	0	0	0	0	0	0	0	0	0	63	74	487	11	0	0	0	0
16	0	0	0	0	0	0	0	0	0	106	0	0	0	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.25b



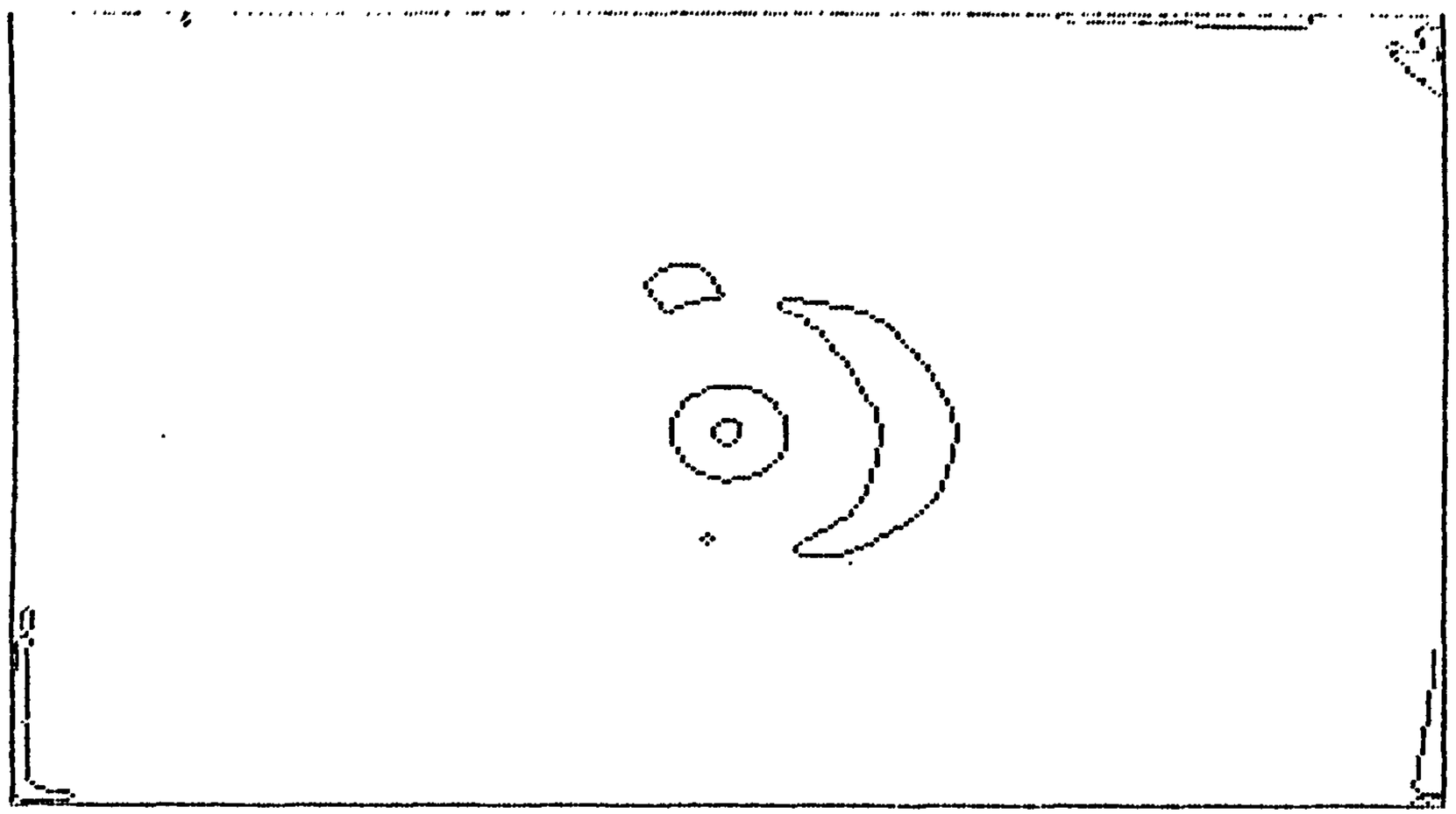


Figure 12.26a

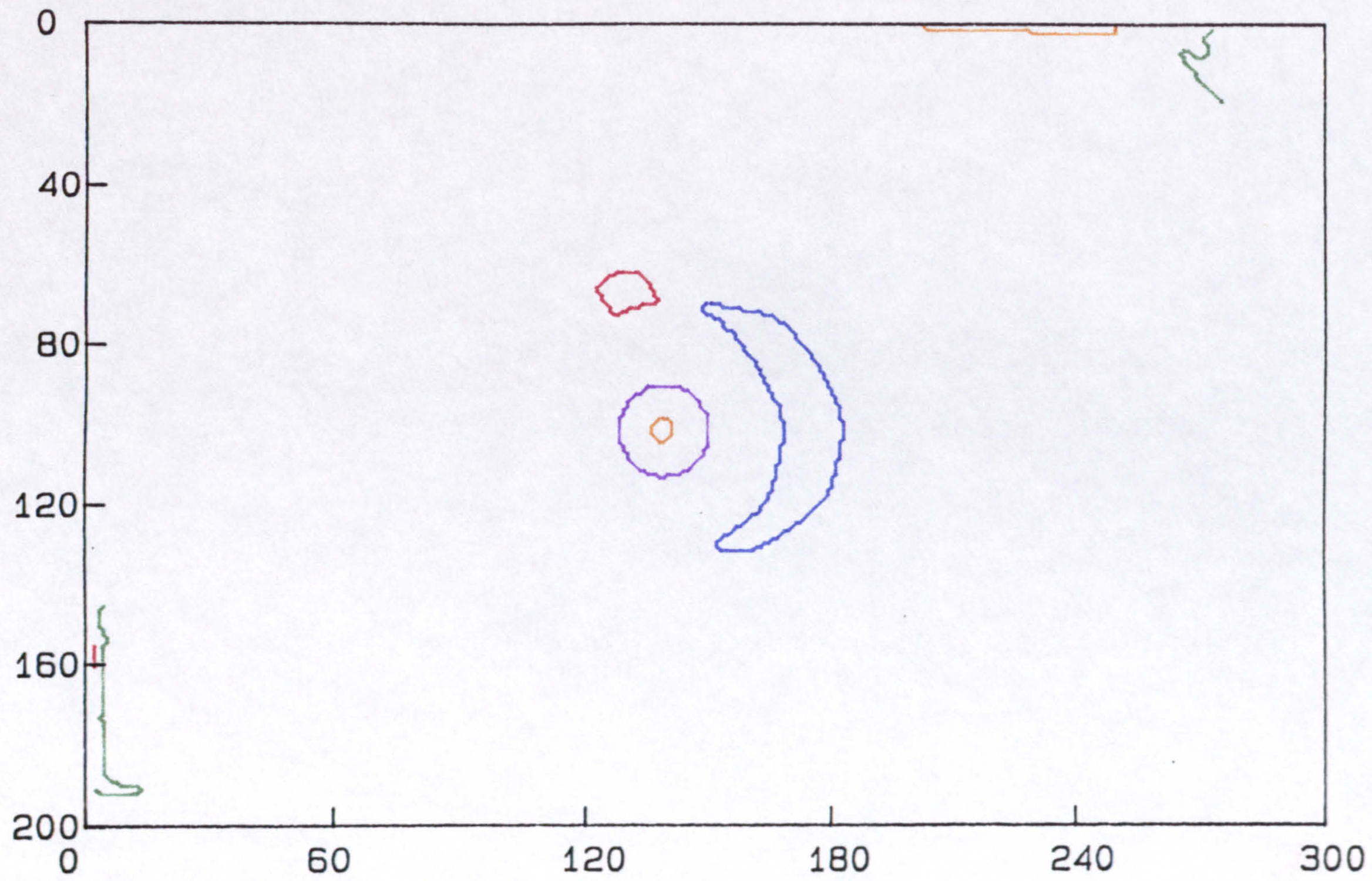
Print of current image being worked on.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
1	0	0	0	0	0	0	0	0	0	51	0	0	0	0	0	0	0
8	149	71	2	169	99	33	154	127	64	157	181	885	37	34	31	56	65
9	0	0	0	0	0	0	0	0	0	64	74	490	11	0	0	0	0
12	0	0	0	0	0	0	0	0	0	106	0	0	0	0	0	0	0

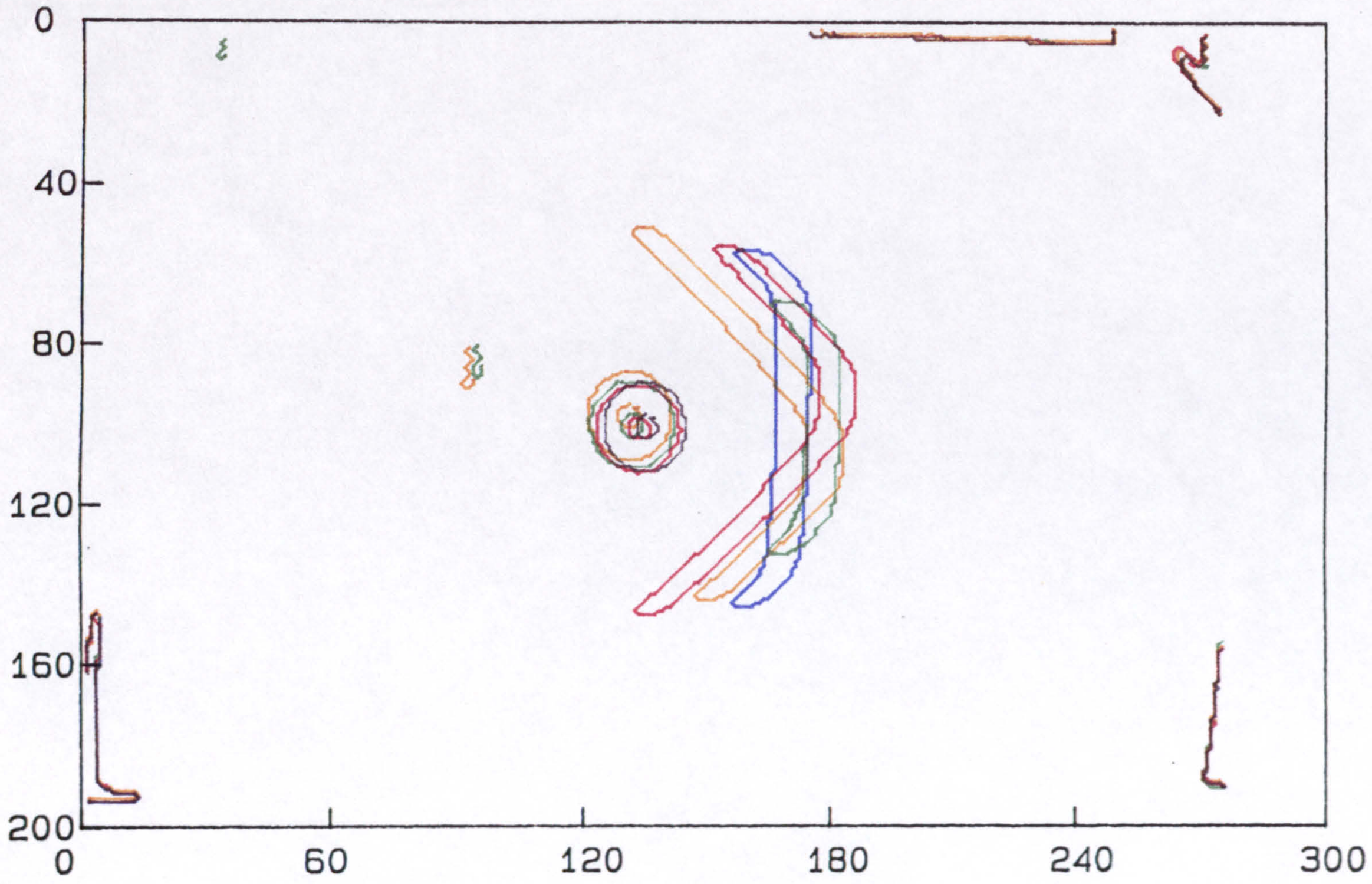
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.26b



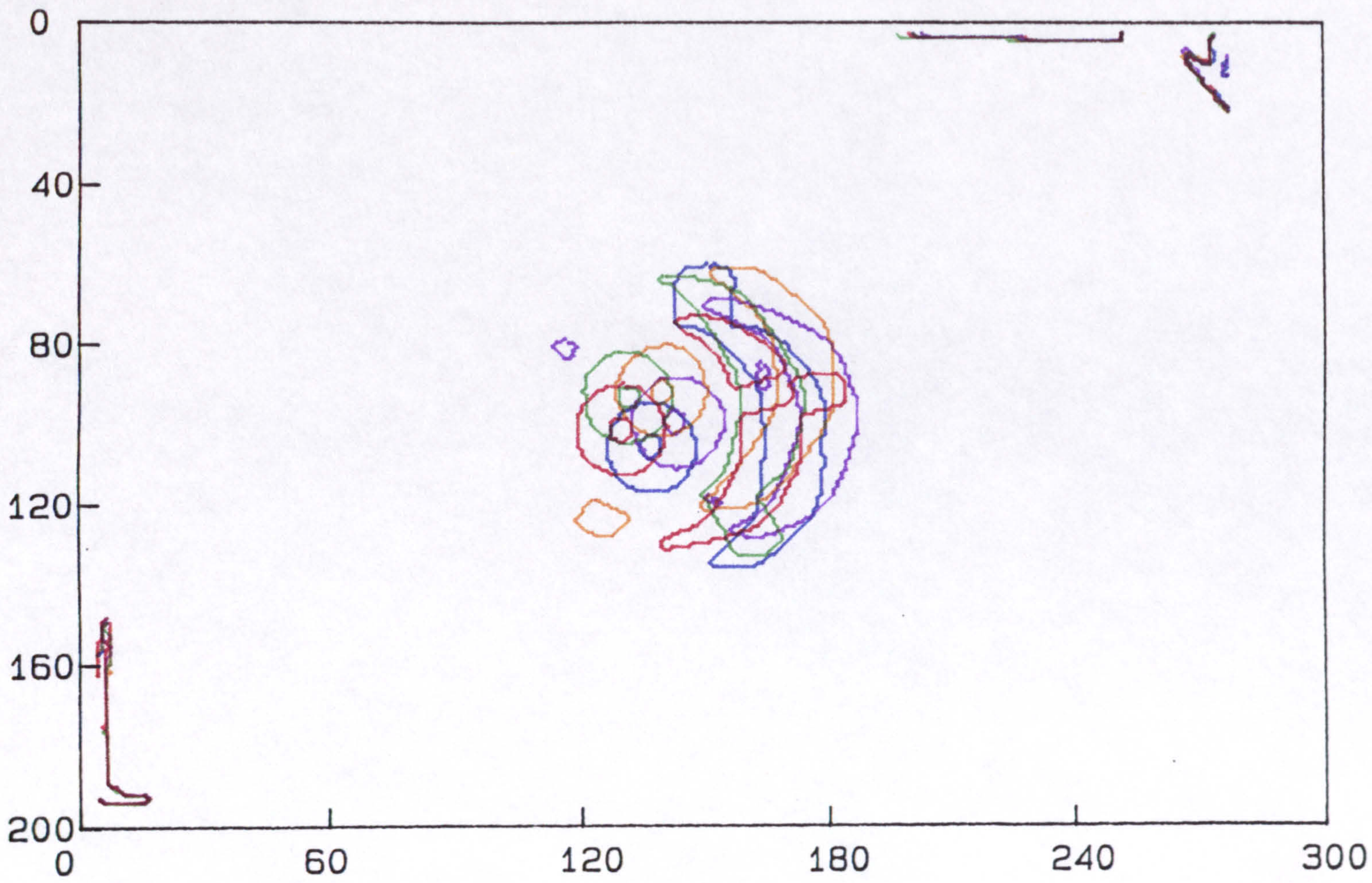
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.27



PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.28



Bowl Minpoint to Pick-up Point as a Function of Angle of Rotation.

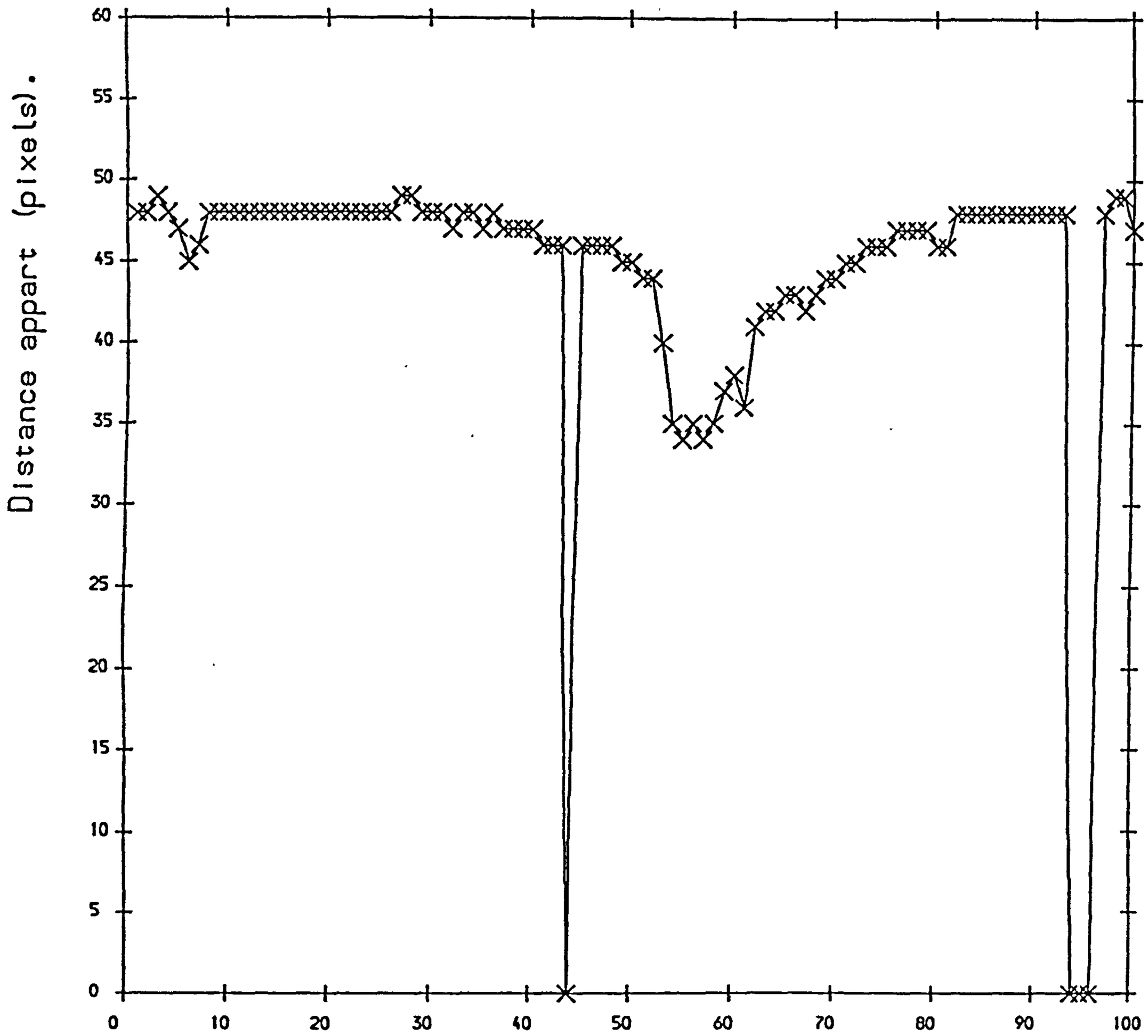


Figure 12.29

Bowl Midpoint to Pick-up Point as a Function of Angle of Rotation.

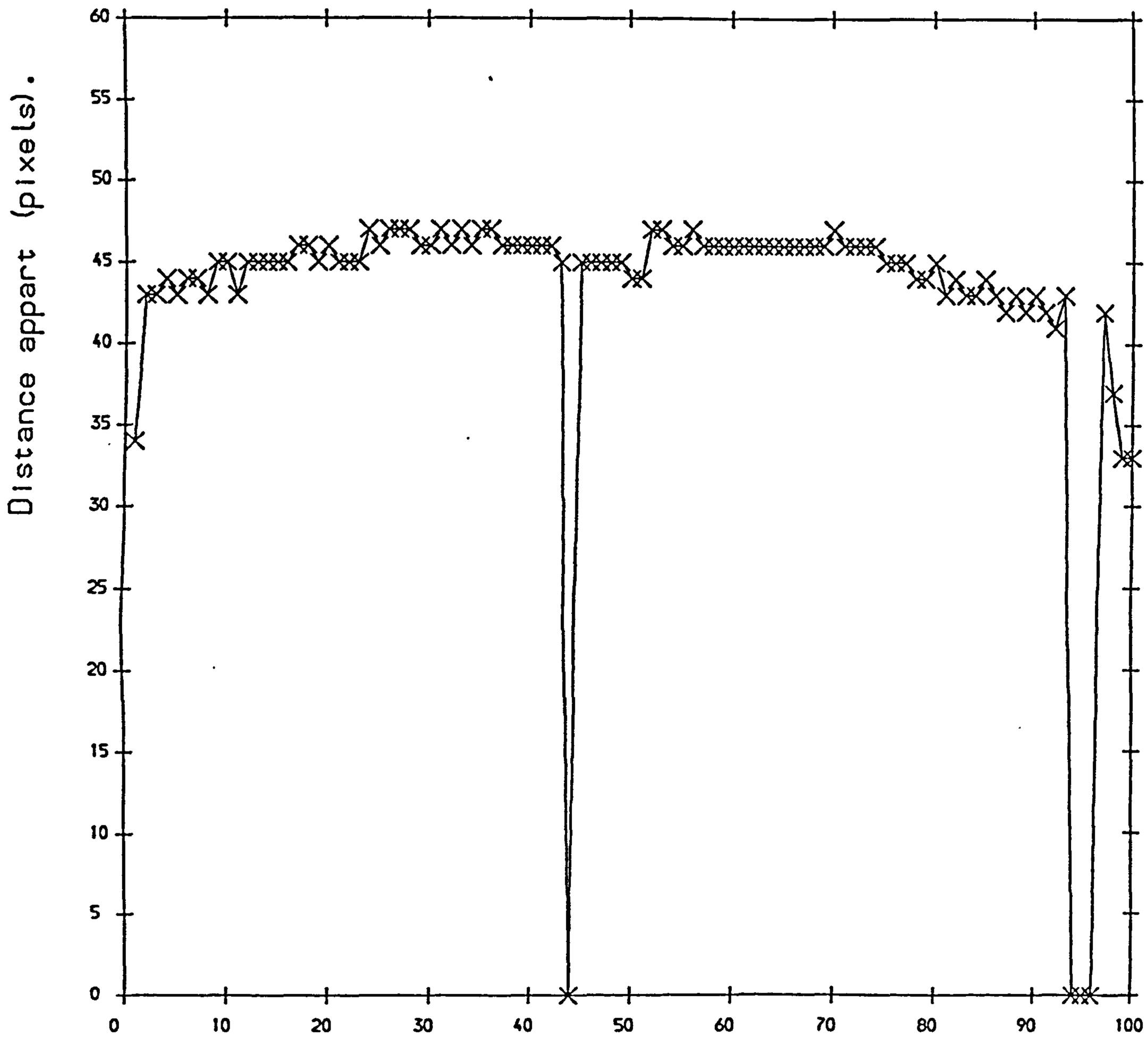


Figure 12.30

Bowl Maxpoint to Pick-up Point as a Function of Angle of Rotation.

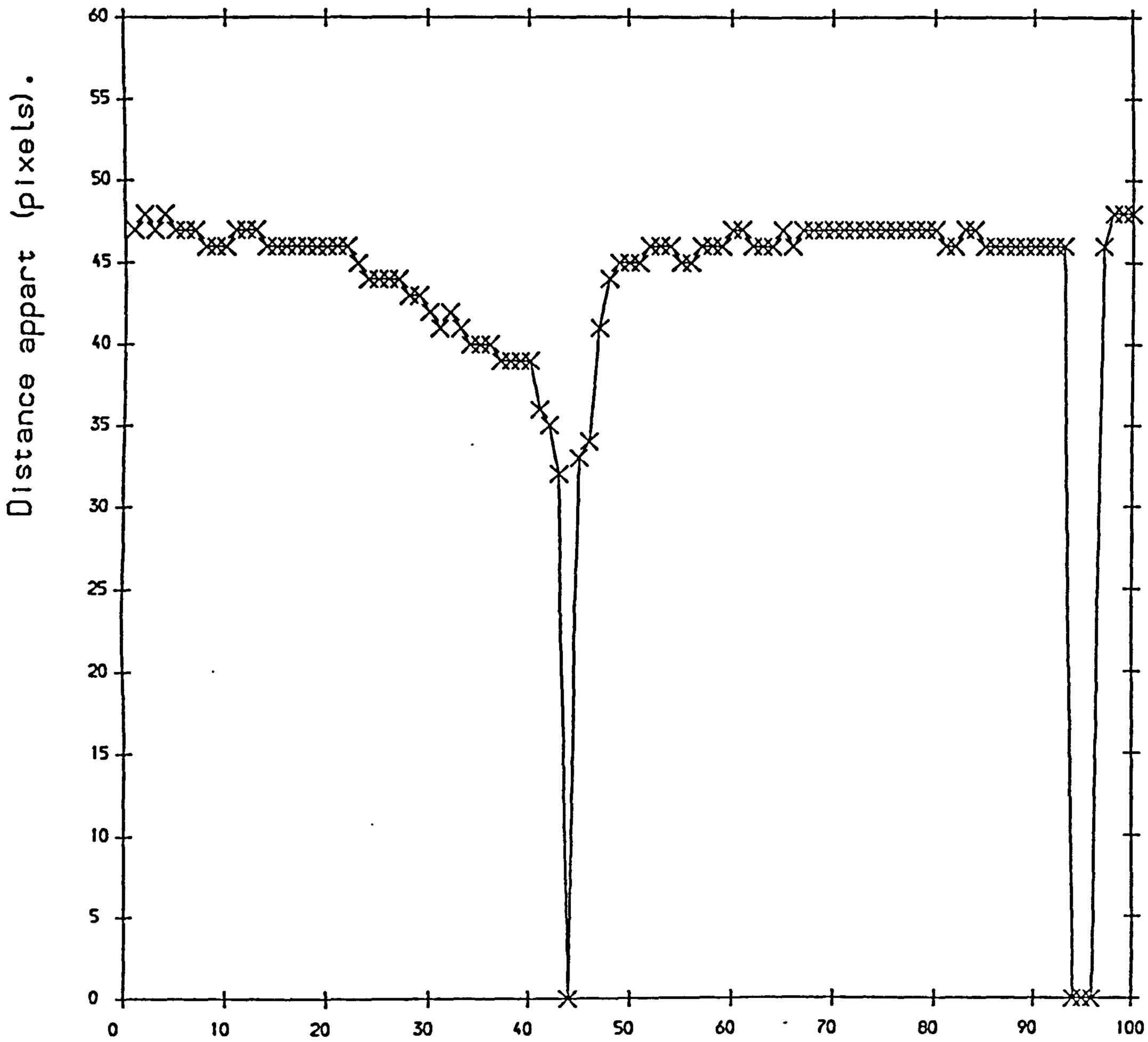


Figure 12.31

Cup Minpoint to Pick-up Point as a Function of Angle of Rotation.

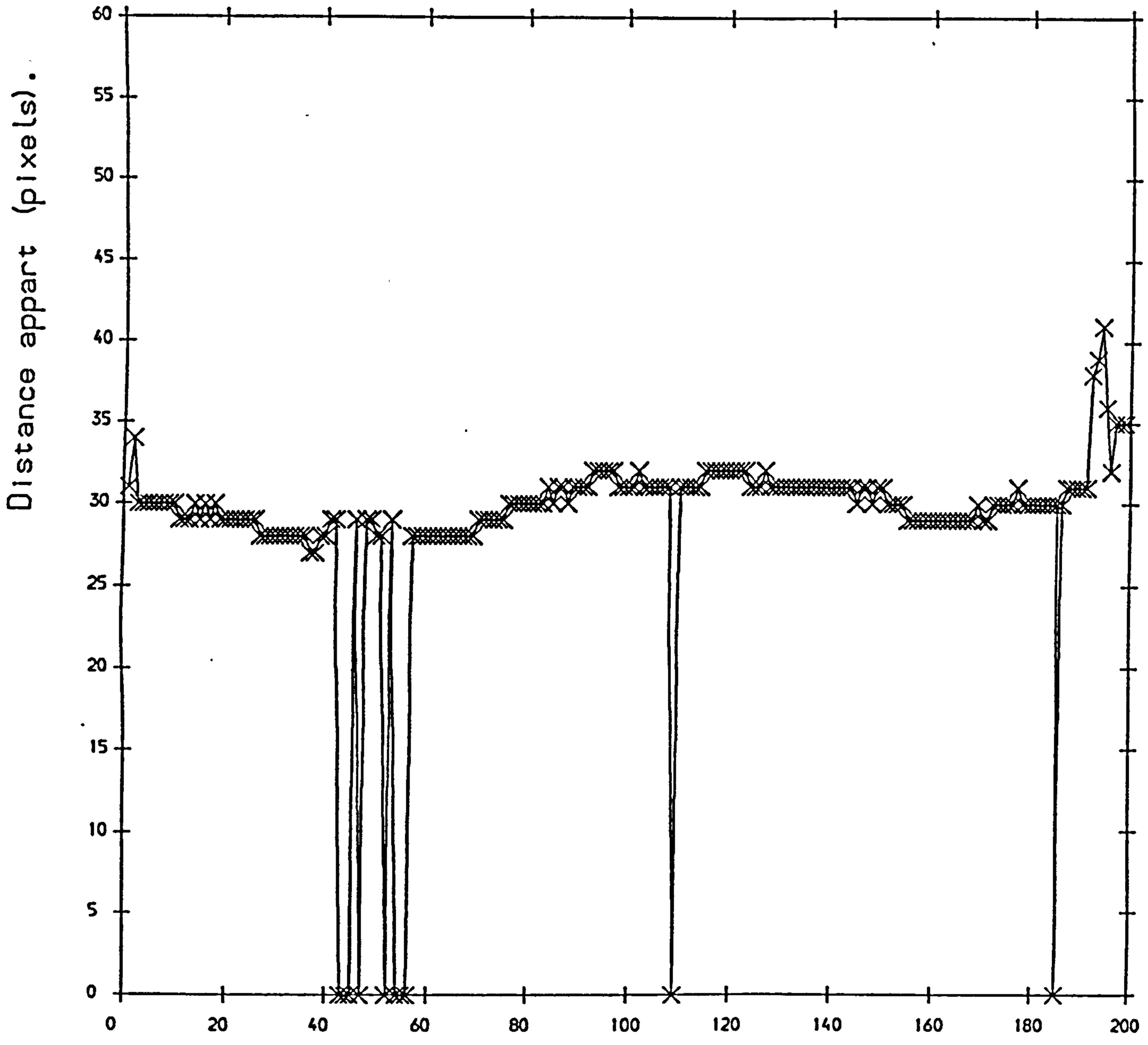


Figure 12.32

Cup Midpoint to Pick-up Point as a Function of Angle of Rotation.

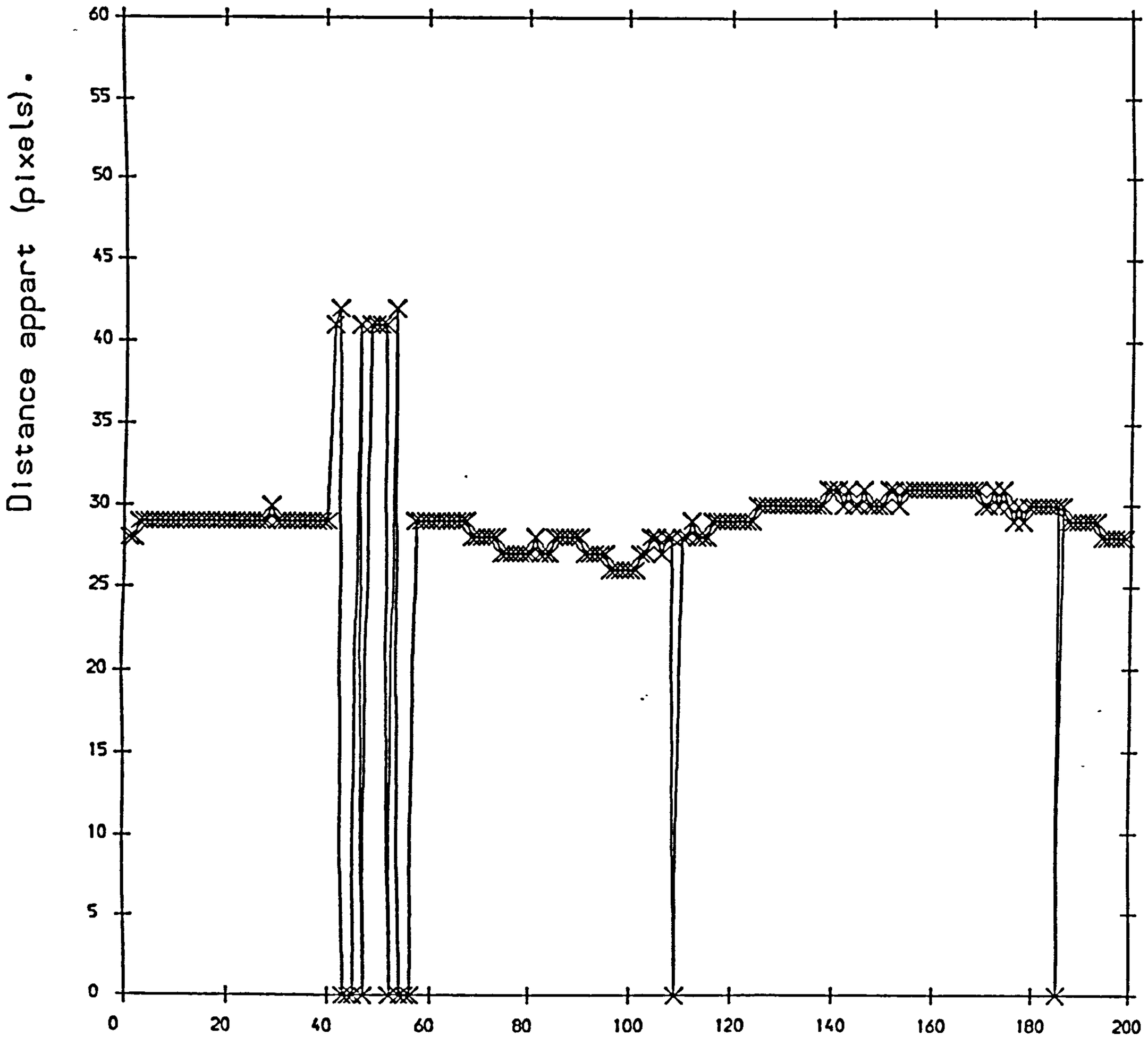


Figure 12.33

Cup Maxpoint to Pick-up Point as a Function of Angle of Rotation.

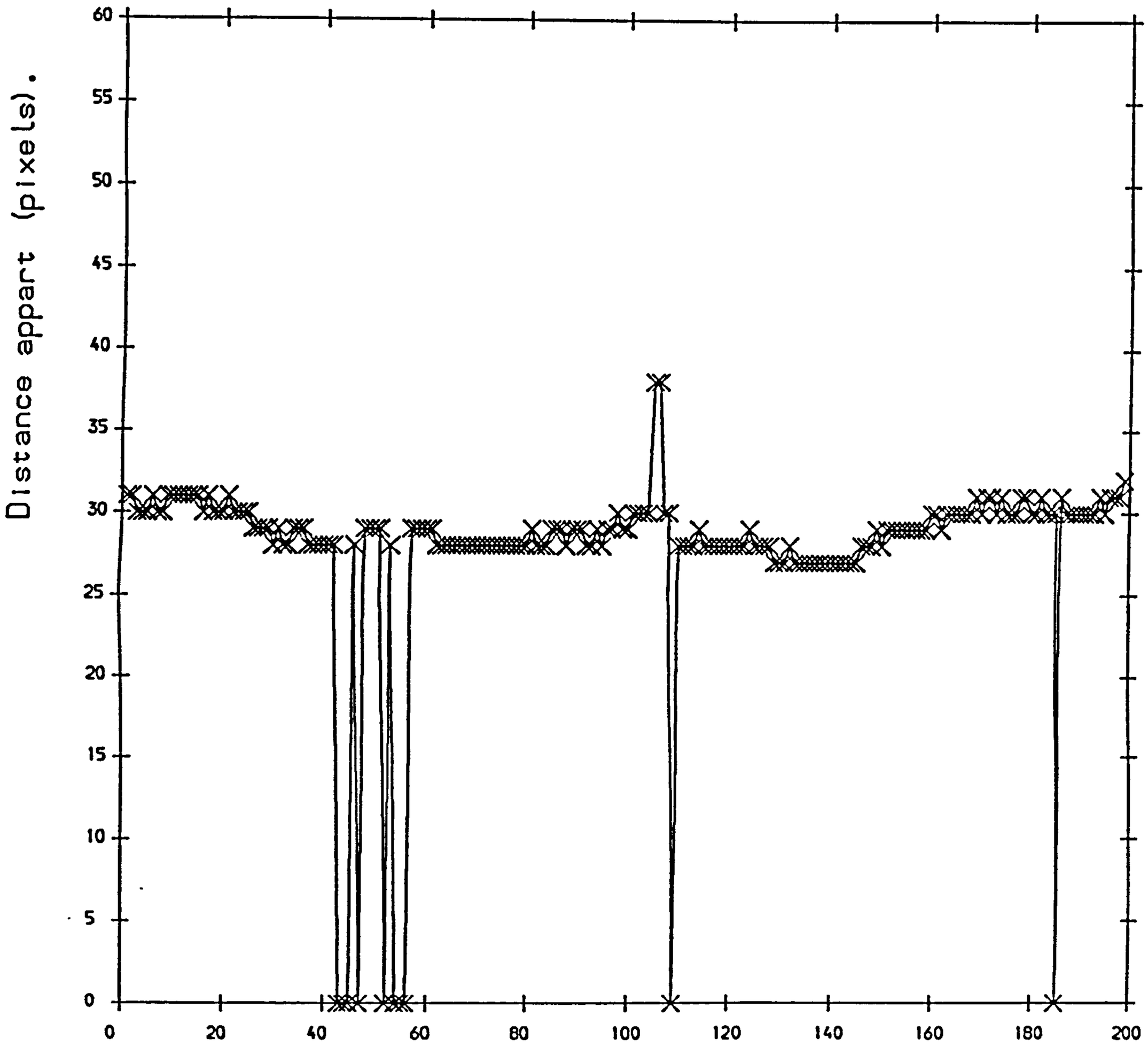


Figure 12.34

Figure 12.35a

```
**************************************************************
*                                                            *
*                          SAPHIRE                           *
*                                                            *
*   The VERSATILE, UNSUPERVISED INDUSTRIAL VISION SYSTEM     *
*                                                            *
*                          from the                          *
*                                                            *
*          DURHAM UNIVERSITY SCHOOL of ENGINEERING           *
*                                                            *
*                       and APPLIED SCIENCE                  *
*                                                            *
**************************************************************
```

```
**************************************************************
*                                                            *
*                  PROGRAM SET-UP PHASE                      *
*                                                            *
**************************************************************
```

Program Parameter Set-up.

Enter the tolerances for features in the cluster-mapping stage.
Enter them in the order: compaction, minmax, minmid, midmax.
The tolerances for feature variation are: 1, 1, 1, 1.

Program Display Option Set-up.

Do you want the maximum speed option to be set?
Do you want read in cluster data printed out?
Is a screen presentation of cluster-data file clusters required.
Is a clustermap test required.

Are the images to be read in from disk or from the Apple?

Enter 1 if the images are to be read in from disk drive 0.
Enter 2 if the images are to be read in from the Apple.

The images are to be read in from disk file.

Is a dot-matrix printout of initial image required?
A dot-matrix printout of original image will be produced.

Is a dot-matrix printout of the thinned/edge-extracted outline required.
A dot-matrix printout of edge-extracted image will be produced.

Is a printout of the feature-table data required?
A feature-table printout will be provided.

Is a presentation of the chain-coded outlines required on the DUET?

Is a presentation of the intercept circles required on the DUET?

Are the pick-up points of identified objects to be plotted on the DUET?

Is a hardcopy of the identification results required?
A hardcopy of identification details will be provided on the printer.

Program operating parameters and input/output options set.

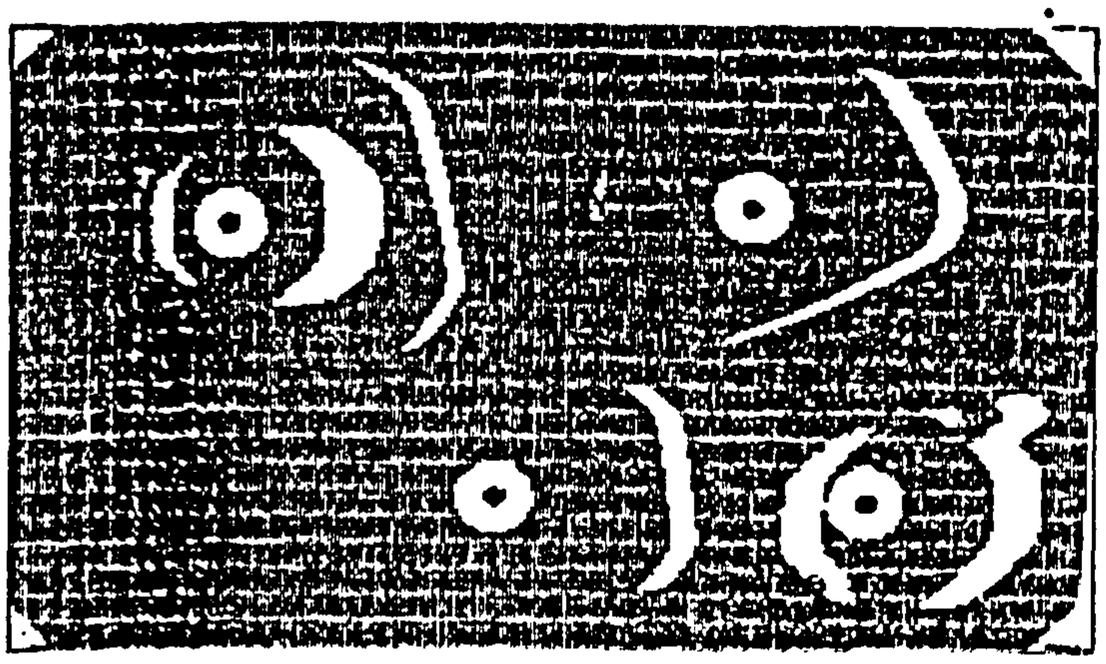
Text cut off in original

```
code= 45.
New cluster ci=6.
code= 46.
No cluster found, new one created, colindex= 15.
New cluster ci=7.
code= 47.
New cluster ci=8.
code= 48.
New cluster ci=9.
code= 49.
New cluster ci=10.
code= 4a.
New cluster ci=11.
code= 4b.
New cluster ci=12.
code= 4c.
No cluster found, new one created, colindex= 16.
New cluster ci=13.
code= 4d.
New cluster ci=0.
code= 60.
No cluster found, new one created, colindex= 17.
No cluster found, new one created, colindex= 18.
New cluster ci=1.
code= 61.
New cluster ci=2.
code= 62.
No cluster found, new one created, colindex= 19.
New cluster ci=3.
code= 63.
No cluster found, new one created, colindex= 20.
New cluster ci=4.
code= 64.
No cluster found, new one created, colindex= 21.
New cluster ci=5.
code= 65.
New cluster ci=6.
code= 66.
No cluster found, new one created, colindex= 22.
New cluster ci=7.
code= 67.
No cluster found, new one created, colindex= 23.
Mapping operation complete. Number of clashes is 24.
Collision index is 0. Clusters colliding are:-
 20, 25,
Collision index is 1. Clusters colliding are:-
 24, 26,
Collision index is 2. Clusters colliding are:-
 26, 29,
Collision index is 3. Clusters colliding are:-
 25, 29,
Collision index is 4. Clusters colliding are:-
 29, 2a,
Collision index is 5. Clusters colliding are:-
 26, 2a,
Collision index is 6. Clusters colliding are:-
 25, 2a,
Collision index is 7. Clusters colliding are:-
 22, 2a,
Collision index is 8. Clusters colliding are:-
 25, 2b,
Collision index is 9. Clusters colliding are:-
 2a, 2b,
Collision index is 10. Clusters colliding are:-
 2b, 2c,
Collision index is 11. Clusters colliding are:-
 23, 2c,
Collision index is 12. Clusters colliding are:-
 2a, 40,
Collision index is 13. Clusters colliding are:-
 25, 40,
Collision index is 14. Clusters colliding are:-
 2b, 40,
Collision index is 15. Clusters colliding are:-
 22, 46,
Collision index is 16. Clusters colliding are:-
 44, 4c,
Collision index is 17. Clusters colliding are:-
 44, 60,
Collision index is 18. Clusters colliding are:-
 4c, 60,
Collision index is 19. Clusters colliding are:-
 43, 62,
Collision index is 20. Clusters colliding are:-
 44, 63,
Collision index is 21. Clusters colliding are:-
 43, 64,
Collision index is 22. Clusters colliding are:-
 44, 66,
Collision index is 23. Clusters colliding are:-
 61, 67,

Cluster data look-up table created.

Object class look-up table generated.
```

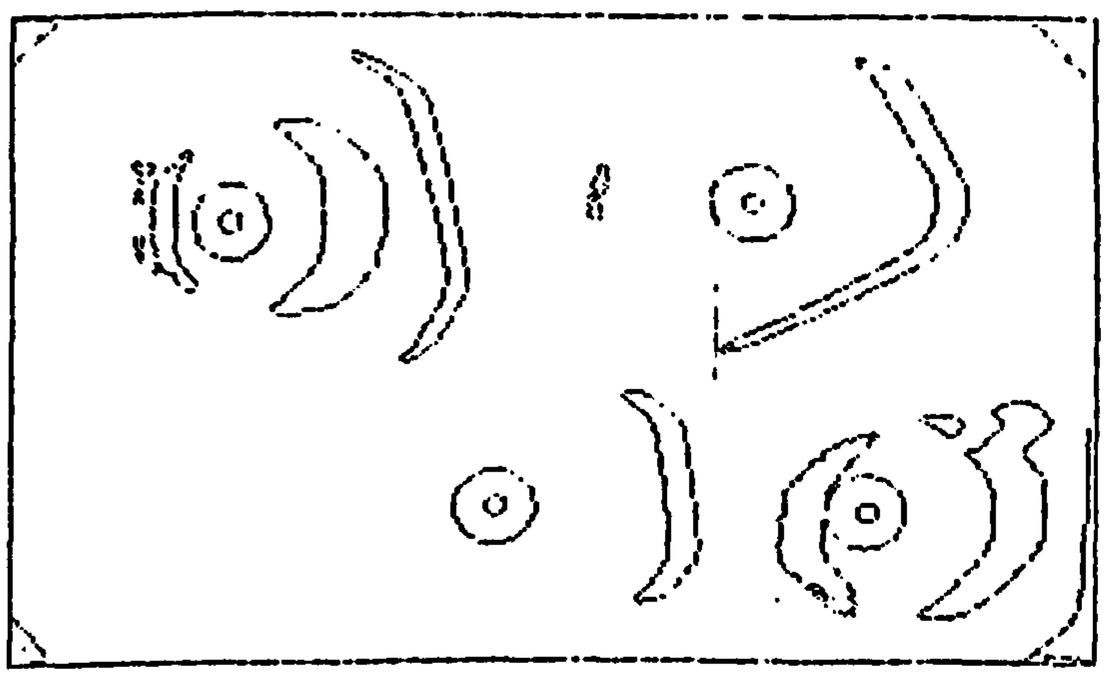
Figure 12.35a Continued.

Figure 12.35a Continued.

```
**********************************************************
*                                                        *
*          DISK FILE INPUT PARAMETERS STAGE              *
*                                                        *
**********************************************************

Enter the number of file images to be accessed on disc.
Enter the base name of the first file in the sequence.
Enter the start index of file with this base name.
Enter the file interval index.
Accessing 3 files from file test__________, with base index 7,
at 1 unit intervals.
File name generated is test__7________.Drive 0 deselected.
Disc image received.

Expansion of bits into bytes completed.
```



```
 Print of current image being worked on.

Edge extraction completed.
```



```
 Print of current image being worked on.

.....
Chain coding sequence completed.

Test for closed outlines completed.

Artificial end of boundary generation completed.

Open line feature point derivation completed.

Open line feature calculation completed.

Closed boundary perimeter calculation completed.
Closed boundary area calculation completed.
Area correction routine entered.
Lateral area correction completed.
Closed boundary compaction-factor calculation completed.
```

Figure 12.35a Continued.

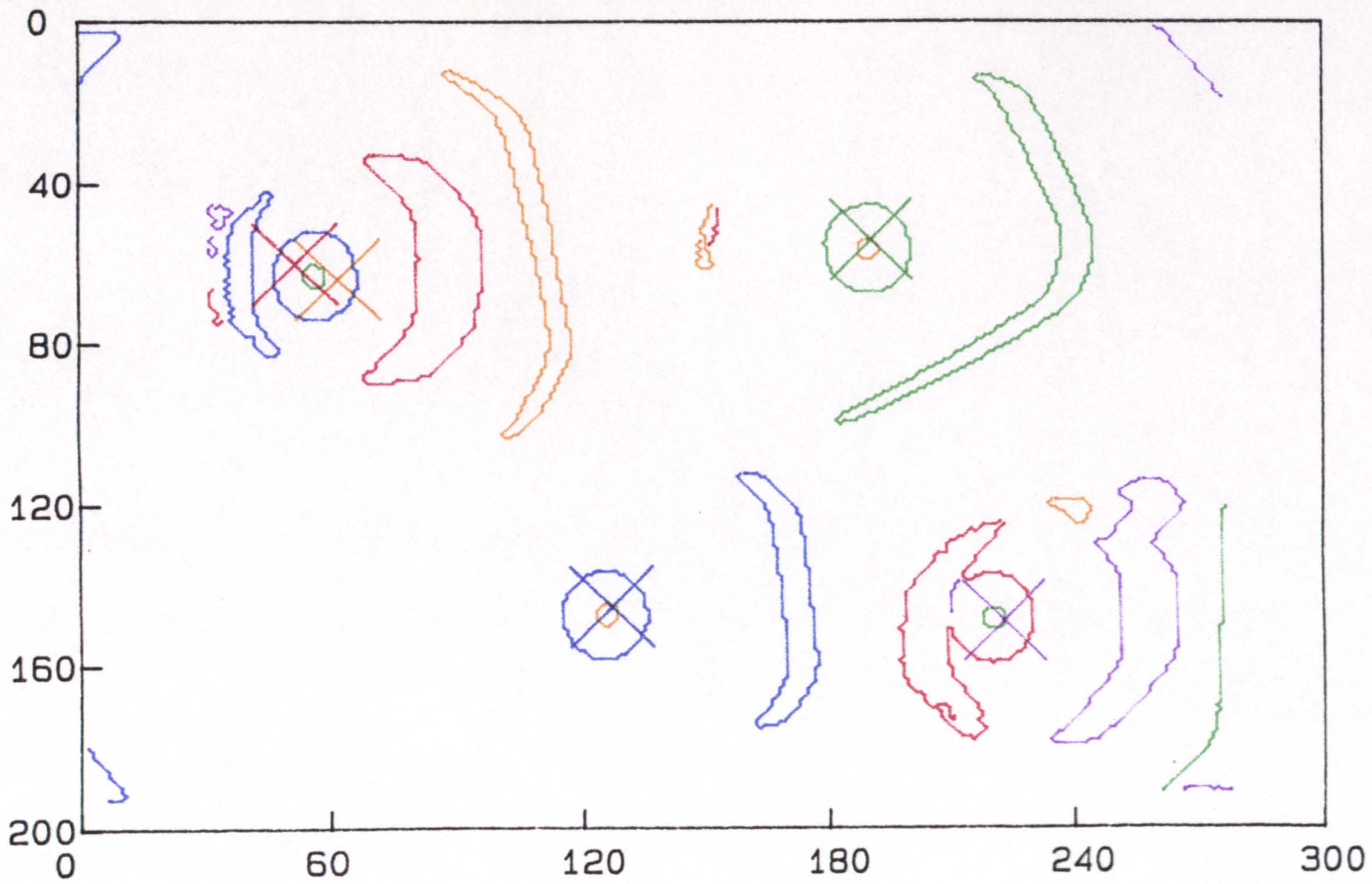
```
Object identification by mapping into the look-up table.

Testing data set 3.

The features for object 3 are:- cmp= 109, mnx= 89, mnd= 68, mdx= 76.
Element in Look-up table is 60.

This object corresponds to file 2, cluster 0.
The description of this file is "plate8*7_".
The number of test images in the object cluster this belongs to is:- 20.

These points correspond to angles of orientation from 67 to 86 degrees.
The distances of this object's centre from the three end points are: 57, 52, 53.

Testing data set 4.

The features for object 4 are:- cmp= 98, mnx= 89, mnd= 42, mdx= 70.
Element in Look-up table is 43.

This object corresponds to file 1, cluster 3.
The description of this file is "bowl8*7_".
The number of test images in the object cluster this belongs to is:- 63.

These points correspond to angles of orientation from 16 to 72 degrees.
The distances of this object's centre from the three end points are: 48, 46, 43.

Testing data set 6.

The features for object 6 are:- cmp= 30, mnx= 52, mnd= 19, mdx= 38.
Element in Look-up table is 26.

This object corresponds to file 0, cluster 6.
The description of this file is "cup8*7__".
The number of test images in the object cluster this belongs to is:- 44.

These points correspond to angles of orientation from 24 to 275 degrees.
The distances of this object's centre from the three end points are: 31, 29, 32.

Testing data set 26.

The features for object 26 are:- cmp= 50, mnx= 59, mnd= 37, mdx= 23.
Element in Look-up table is 40.

This object corresponds to file 1, cluster 0.
The description of this file is "bowl8*7_".
The number of test images in the object cluster this belongs to is:- 11.

These points correspond to angles of orientation from 81 to 90 degrees.
The distances of this object's centre from the three end points are: 45, 44, 45.

Testing data set 27.

The features for object 27 are:- cmp= 34, mnx= 48, mnd= 13, mdx= 39.
Element in Look-up table is 24.

This object corresponds to file 0, cluster 4.
The description of this file is "cup8*7__".
The number of test images in the object cluster this belongs to is:- 28.

These points correspond to angles of orientation from 40 to 66 degrees.
The distances of this object's centre from the three end points are: 30, 30, 30.

Testing data set 30.

The features for object 30 are:- cmp= 34, mnx= 20, mnd= 33, mdx= 20.
Element in Look-up table is 0.

No match with learnt objects found for this object.


Object identification stage completed.

Routine to determine intercept points entered.
Intercept point calculation for file= 2, cluster= 0.
Intercept point calculation for file= 1, cluster= 3.
Intercept point calculation for file= 0, cluster= 6.
Intercept point calculation for file= 1, cluster= 0.
Intercept point calculation for file= 0, cluster= 4.
Intersection point of endpoint radii calculated.

Intersect triangle centre of gravity routine entered.
For cross index=0, x centre= 62, y centre= 62.
For cross index=1, x centre= 192, y centre= 53.
For cross index=2, x centre= 52, y centre= 58.
For cross index=3, x centre= 127, y centre= 144.
For cross index=4, x centre= 224, y centre= 148.
Centre of gravity of intersection radii calculated.
```

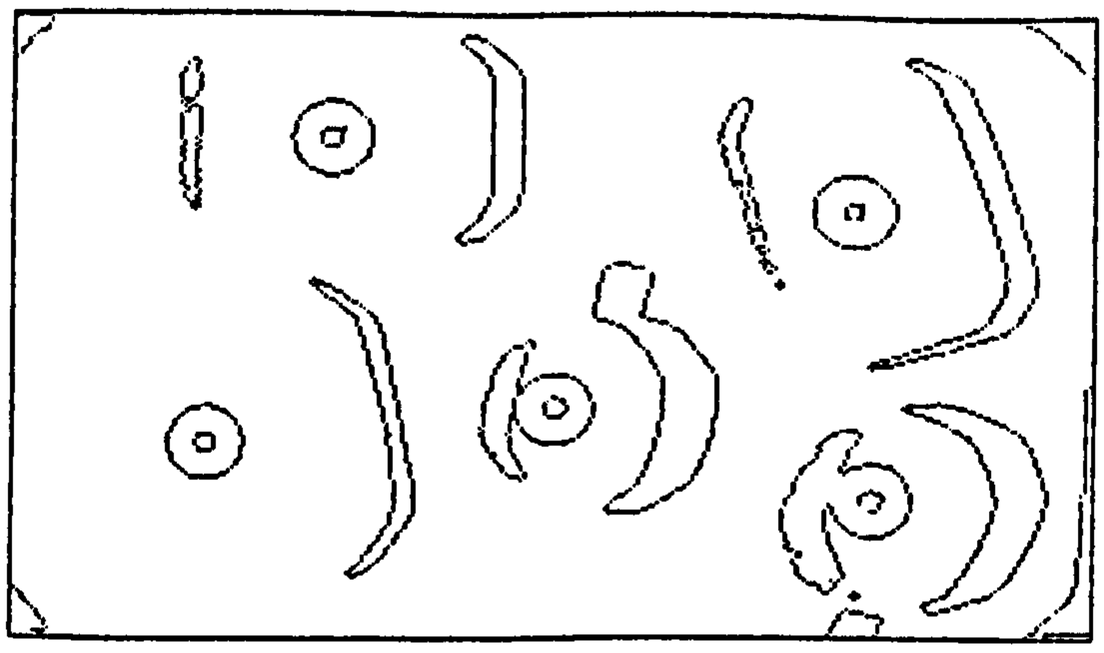
TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
3	88	11	1	113	75	67	103	99	91	194	220	441	109	68	26	89	94
4	218	13	1	239	50	39	185	96	106	230	272	754	98	42	70	89	112
6	69	33	2	81	48	17	70	85	54	140	158	819	30	19	38	52	57
8	36	72	33	36	72	33	36	72	33	88	105	261	42	0	0	0	0
11	0	0	0	0	0	0	0	0	0	61	72	415	12	0	0	0	0
16	0	0	0	0	0	0	0	0	0	60	69	399	11	0	0	0	0
26	158	112	1	170	148	37	164	171	60	137	152	457	50	37	23	59	60
27	247	129	19	254	140	30	237	176	67	153	184	986	34	13	39	48	52
29	0	0	0	0	0	0	0	0	0	72	0	0	0	0	0	0	0
30	198	150	36	231	151	121	215	138	146	161	193	1067	34	33	20	20	53
31	0	0	0	0	0	0	0	0	0	59	70	490	10	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.35b





Print of current image being worked on.

Figure 12.36a

```
Object identification by mapping into the look-up table.
Testing data set 2.
The features for object 2 are:- cmp= 47, mnx= 59, mnd= 13, mdx= 48.
Element in Look-up table is 46.

This object corresponds to file 1, cluster 6.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 10.

These points correspond to angles of orientation from 91 to 99 degrees.
The distances of this object's centre from the three end points are: 44, 46, 46.

Testing data set 4.

The features for object 4 are:- cmp= 92, mnx= 95, mnd= 69, mdx= 45.
Element in Look-up table is 44.

This object corresponds to file 1, cluster 4.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 46.

These points correspond to angles of orientation from 121 to 165 degrees.
The distances of this object's centre from the three end points are: 46, 43, 46.

Testing data set 19.

The features for object 19 are:- cmp= 40, mnx= 57, mnd= 26, mdx= 40.
Element in Look-up table is cff.

A cluster collision has occured at this point.
The collision index of this collision is 12. The clusters colliding are:-
file 0, cluster 10.
The file description is cup8*7_.
The number of test images in the object cluster this belongs to is:- 127.
These points correspond to angles of orientation from 1 to 360 degrees.
The distances of the pick-up point from the three end points are: 31, 28, 30.

file 1, cluster 0.
The file description is bowl8*7_.
The number of test images in the object cluster this belongs to is:- 11.
These points correspond to angles of orientation from 81 to 90 degrees.
The distances of the pick-up point from the three end points are: 45, 44, 45.


Testing data set 20.

The features for object 20 are:- cmp= 126, mnx= 87, mnd= 64, mdx= 27.
Element in Look-up table is 60.

This object corresponds to file 2, cluster 0.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 20.

These points correspond to angles of orientation from 67 to 86 degrees.
The distances of this object's centre from the three end points are: 57, 52, 53.

Testing data set 28.

The features for object 28 are:- cmp= 38, mnx= 59, mnd= 36, mdx= 36.
Element in Look-up table is 3ff.

A cluster collision has occured at this point.
The collision index of this collision is 3. The clusters colliding are:-
file 0, cluster 5.
The file description is cup3*7_.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 133 to 356 degrees.
The distances of the pick-up point from the three end points are: 31, 29, 31.

file 0, cluster 9.
The file description is cup3*7_.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 22 to 323 degrees.
The distances of the pick-up point from the three end points are: 31, 30, 29.


Testing data set 30.

The features for object 30 are:- cmp= 31, mnx= 29, mnd= 34, mdx= 20.
Element in Look-up table is 0.

No match with learnt objects found for this object.
```

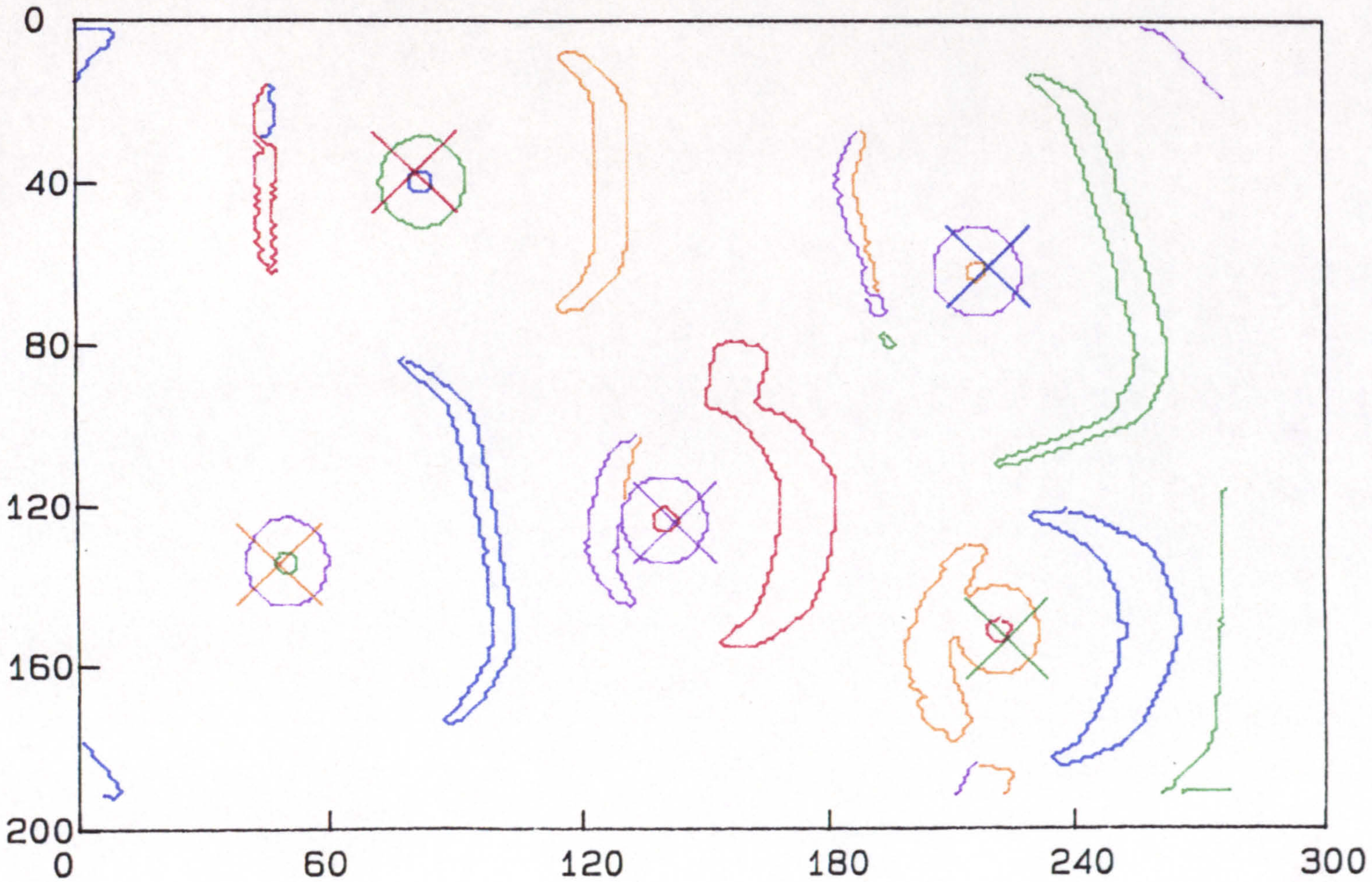
Figure 12.36a Continued.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
2	116	8	2	124	19	13	117	67	61	138	150	473	47	13	48	59	61
4	231	13	1	257	77	66	224	108	113	244	277	834	92	69	45	95	114
7	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	60	71	390	12	0	0	0	0
12	0	0	0	0	0	0	0	0	0	61	71	422	11	0	0	0	0
19	151	94	17	169	113	40	156	151	79	182	210	1091	40	26	40	57	66
20	77	83	1	99	144	63	89	170	89	189	214	361	126	64	27	87	91
23	0	0	0	0	0	0	0	0	0	128	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0
28	231	122	9	255	149	43	238	181	77	166	195	985	38	36	36	59	72
29	0	0	0	0	0	0	0	0	0	58	68	456	10	0	0	0	0
30	203	169	43	233	153	107	216	142	132	149	176	982	31	34	20	29	54

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.36b



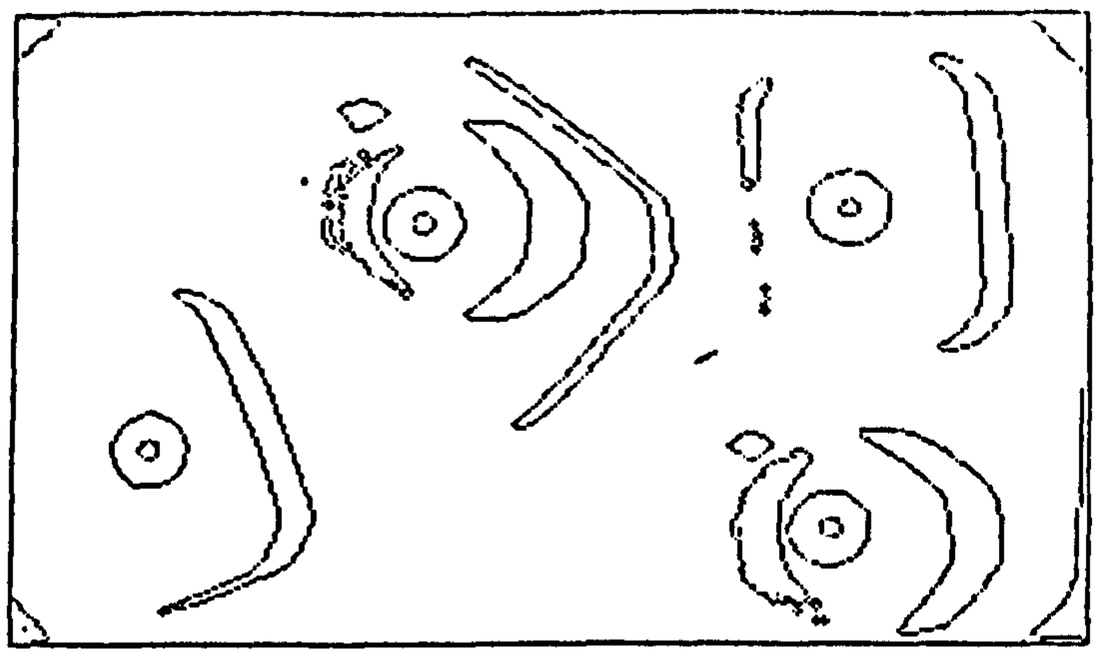


Figure 12.37a

Print of current image being worked on.

Object identification by mapping into the look-up table.

Testing data set 3.

The features for object 3 are:- cmp= 59, mnx= 88, mnd= 69, mdx= 22.
Element in Look-up table is 41.

This object corresponds to file 1, cluster 1.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 13.

These points correspond to angles of orientation from 166 to 177 degrees.
The distances of this object's centre from the three end points are: 49, 37, 47.

Testing data set 4.

The features for object 4 are:- cmp= 182, mnx= 111, mnd= 69, mdx= 69.
Element in Look-up table is 61.

This object corresponds to file 2, cluster 1.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 97.

These points correspond to angles of orientation from 20 to 155 degrees.
The distances of this object's centre from the three end points are: 57, 55, 55.

Testing data set 8.

The features for object 8 are:- cmp= 34, mnx= 37, mnd= 28, mdx= 37.
Element in Look-up table is 29.

This object corresponds to file 0, cluster 9.
The description of this file is "cup8*7__."
The number of test images in the object cluster this belongs to is:- 37.

These points correspond to angles of orientation from 22 to 323 degrees.
The distances of this object's centre from the three end points are: 31, 30, 29.

Testing data set 31.

The features for object 31 are:- cmp= 107, mnx= 95, mnd= 70, mdx= 41.
Element in Look-up table is 63.

This object corresponds to file 2, cluster 3.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 16.

These points correspond to angles of orientation from 159 to 172 degrees.
The distances of this object's centre from the three end points are: 59, 52, 55.

Testing data set 37.

The features for object 37 are:- cmp= 37, mnx= 58, mnd= 39, mdx= 28.
Element in Look-up table is 8ff.

A cluster collision has occured at this point.
The collision index of this collision is 8. The clusters colliding are:-

file 0, cluster 5.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 133 to 356 degrees.
The distances of the pick-up point from the three end points are: 31, 29, 31.

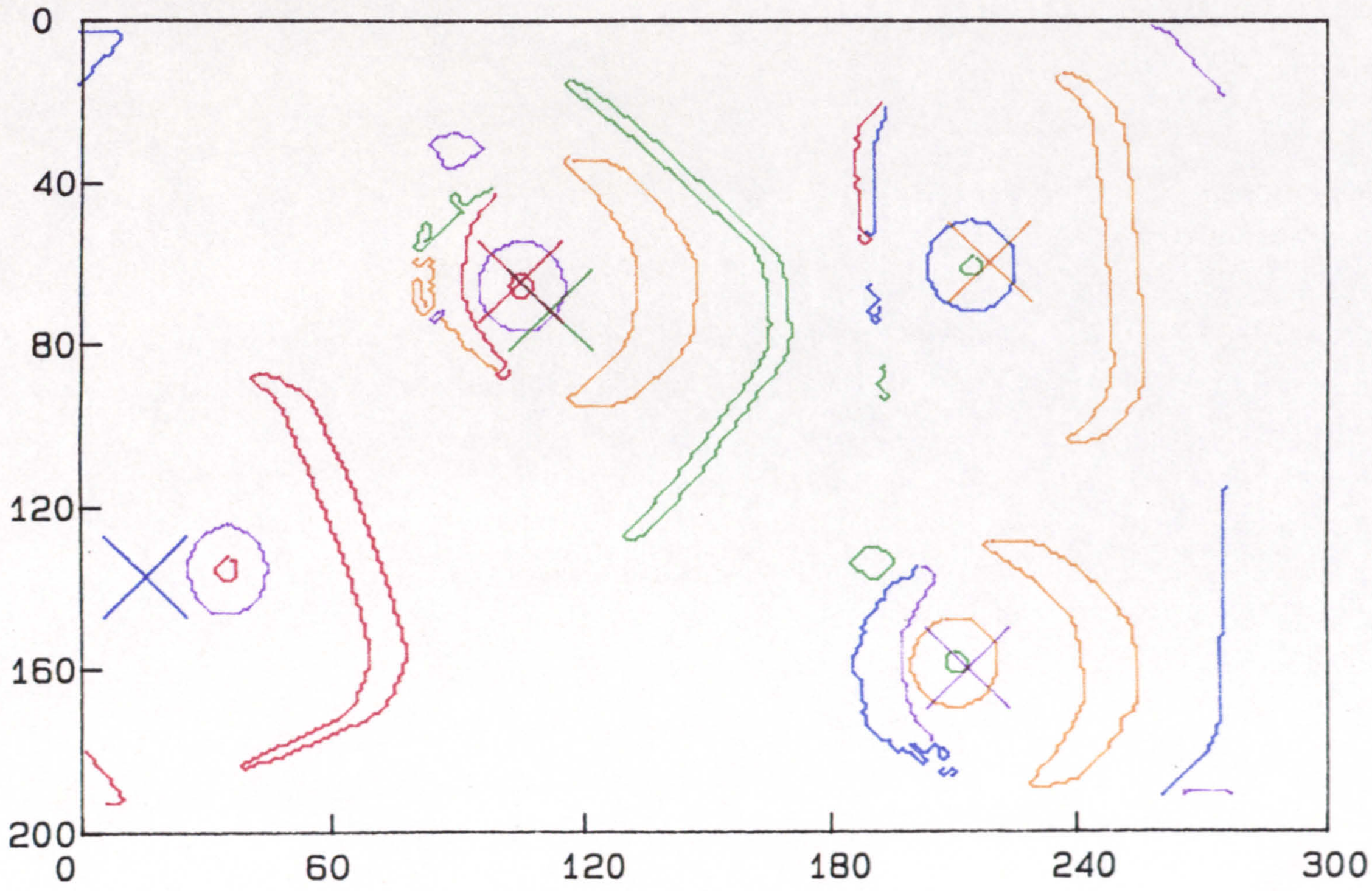
file 0, cluster 11.
The file description is cup9*7__.
The number of test images in the object cluster this belongs to is:- 17.
These points correspond to angles of orientation from 0 to 354 degrees.
The distances of the pick-up point from the three end points are: 33, 28, 31.

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
3	238	14	2	252	82	71	242	102	91	198	214	774	59	69	22	88	91
4	117	14	1	166	63	55	132	124	116	243	308	520	182	69	69	111	138
8	117	34	2	134	57	25	119	91	59	148	173	861	34	28	37	57	65
15	0	0	0	0	0	0	0	0	0	62	72	429	12	0	0	0	0
17	0	0	0	0	0	0	0	0	0	60	70	414	11	0	0	0	0
31	41	86	1	69	151	67	40	181	108	234	270	680	107	70	41	95	111
35	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	58	68	454	10	0	0	0	0
37	219	129	1	244	160	36	232	186	62	152	178	851	37	39	28	58	67
41	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0	0	60	70	308	9	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.37b



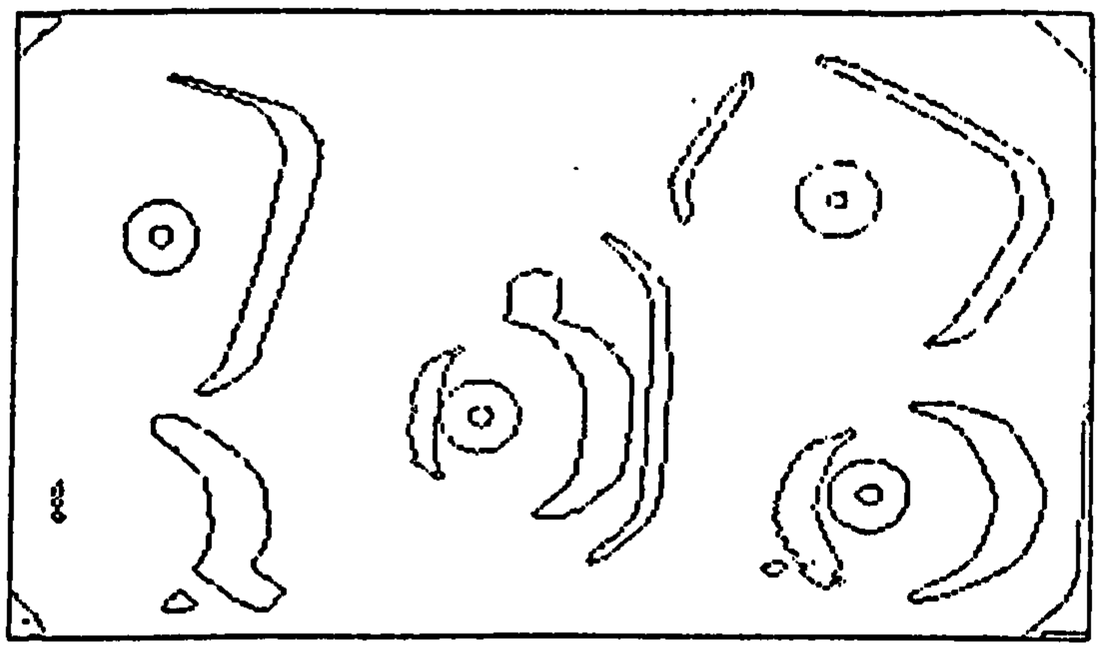


Figure 12.38a

Print of current image being worked on.

Object identification by mapping into the look-up table.
Testing data set 3.
The features for object 3 are:- cmp= 99, mnx= 90, mnd= 67, mdx= 48.
Element in Look-up table is 44.

This object corresponds to file 1, cluster 4.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 46.

These points correspond to angles of orientation from 121 to 165 degrees.
The distances of this object's centre from the three end points are: 46, 43, 46.

Testing data set 5.

The features for object 5 are:- cmp= 100, mnx= 94, mnd= 38, mdx= 75.
Element in Look-up table is 13ff.

A cluster collision has occured at this point.
The collision index of this collision is 19. The clusters colliding are:-

file 1, cluster 3.
The file description is bowl8*7_.
The number of test images in the object cluster this belongs to is:- 63.
These points correspond to angles of orientation from 16 to 72 degrees.
The distances of the pick-up point from the three end points are: 48, 46, 43.

file 2, cluster 2.
The file description is plate8*7.
The number of test images in the object cluster this belongs to is:- 22.
These points correspond to angles of orientation from 87 to 106 degrees.
The distances of the pick-up point from the three end points are: 54, 55, 56.

Testing data set 11.
The features for object 11 are:- cmp= 112, mnx= 96, mnd= 23, mdx= 78.
Element in Look-up table is 64.

This object corresponds to file 2, cluster 4.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 22.

These points correspond to angles of orientation from 0 to 180 degrees.
The distances of this object's centre from the three end points are: 57, 52, 57.

Testing data set 12.

The features for object 12 are:- cmp= 43, mnx= 58, mnd= 29, mdx= 38.
Element in Look-up table is 9ff.

A cluster collision has occured at this point.
The collision index of this collision is 9. The clusters colliding are:-

file 0, cluster 10.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 127.
These points correspond to angles of orientation from 1 to 360 degrees.
The distances of the pick-up point from the three end points are: 31, 28, 30.

file 0, cluster 11.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 17.
These points correspond to angles of orientation from 0 to 354 degrees.
The distances of the pick-up point from the three end points are: 33, 28, 31.

Testing data set 18.

The features for object 18 are:- cmp= 44, mnx= 57, mnd= 35, mdx= 36.
Element in Look-up table is 6ff.

A cluster collision has occured at this point.
The collision index of this collision is 6. The clusters colliding are:-

file 0, cluster 5.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 133 to 356 degrees.
The distances of the pick-up point from the three end points are: 31, 29, 31.

file 0, cluster 10.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 127.
These points correspond to angles of orientation from 1 to 360 degrees.
The distances of the pick-up point from the three end points are: 31, 28, 30.

Testing data set 20.

The features for object 20 are:- cmp= 29, mnx= 42, mnd= 31, mdx= 13.
Element in Look-up table is 20.

This object corresponds to file 0, cluster 0.
The description of this file is "cup8*7__."
The number of test images in the object cluster this belongs to is:- 32.

These points correspond to angles of orientation from 107 to 133 degrees.
The distances of this object's centre from the three end points are: 28, 30, 30.

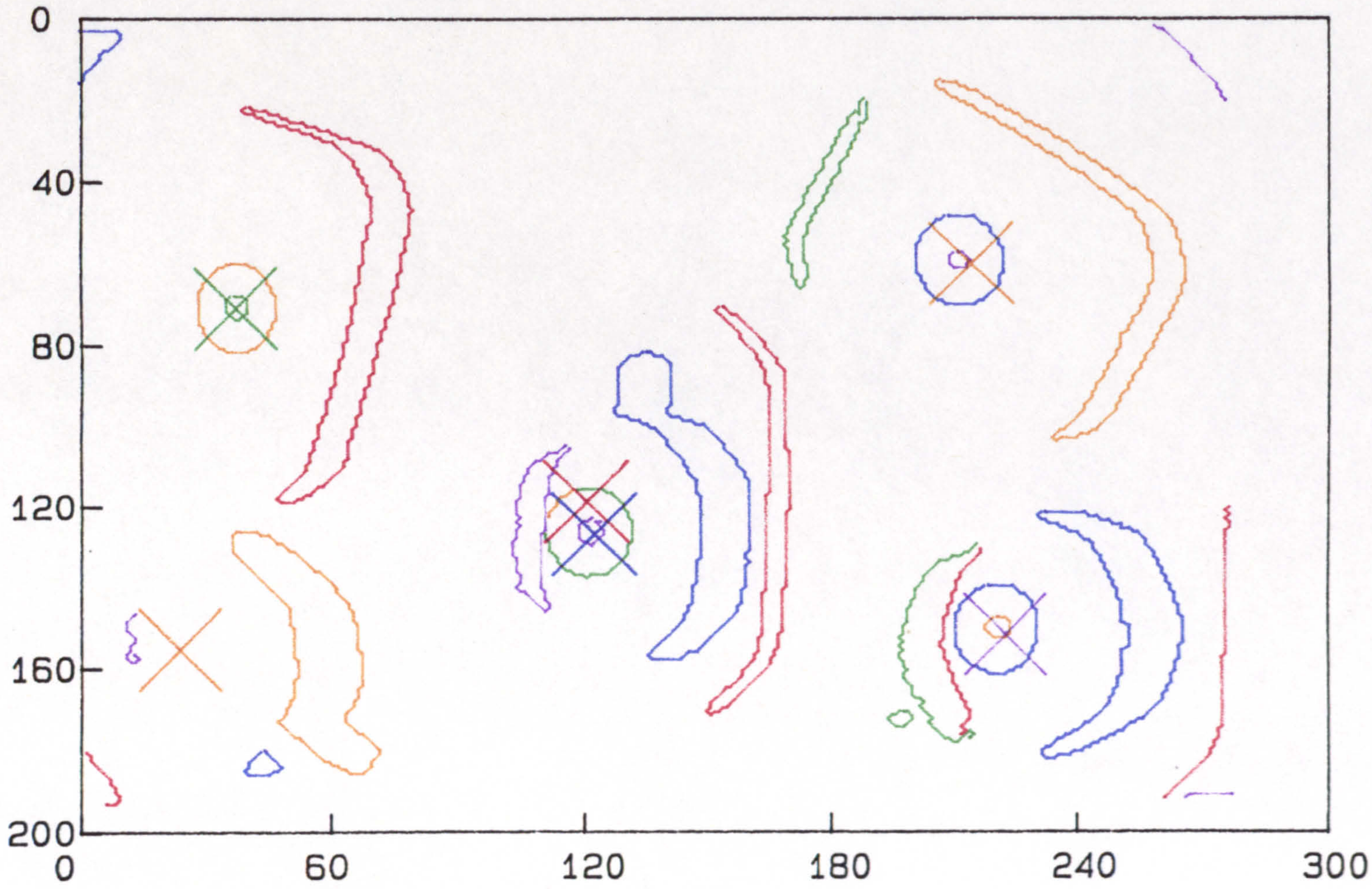
TABLE of IMAGE FEATURES

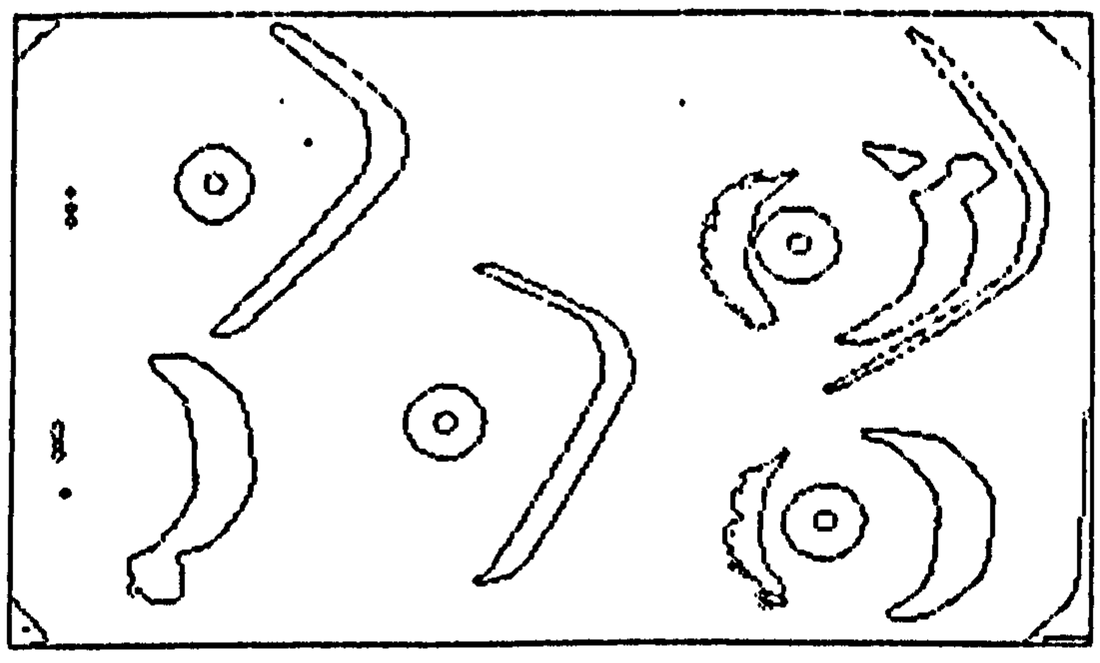
Figure 12.38a Continued.

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
3	208	15	2	261	57	61	238	100	105	226	272	744	99	67	48	90	115
4	170	54	38	170	54	38	170	54	38	97	115	211	62	0	0	0	0
5	39	20	1	70	42	37	49	114	109	237	268	714	100	38	75	94	113
6	0	0	0	0	0	0	0	0	0	62	71	422	11	0	0	0	0
9	0	0	0	0	0	0	0	0	0	58	67	392	11	0	0	0	0
11	153	70	1	166	89	20	152	166	98	207	229	467	112	23	78	96	101
12	128	95	17	149	116	44	138	153	81	179	207	989	43	29	38	58	67
13	103	130	30	103	130	30	103	130	30	86	99	330	29	0	0	0	0
17	0	0	0	0	0	0	0	0	0	73	0	0	0	0	0	0	0
18	232	121	1	255	148	35	235	178	69	167	192	823	44	35	36	57	71
20	36	129	5	52	156	32	49	169	45	138	166	949	29	31	13	42	44
22	0	0	0	0	0	0	0	0	0	63	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	61	70	496	9	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.38b





Print of current image being worked on.

Figure 12.39a

```
Object identification by mapping into the look-up table.
Testing data set 2.
The features for object 2 are:- cmp= 86, mnx= 91, mnd= 39, mdx= 71.
Element in Look-up table is 43.

This object corresponds to file 1, cluster 3.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 63.

These points correspond to angles of orientation from 16 to 72 degrees.
The distances of this object's centre from the three end points are: 48, 46, 43.

Testing data set 3.

The features for bbject 3 are:- cmp= 147, mnx= 110, mnd= 59, mdx= 77.
Element in Look-up table is 61.

This object corresponds to file 2, cluster 1.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 97.

These points correspond to angles of orientation from 20 to 155 degrees.
The distances of this object's centre from the three end points are: 57, 55, 55.

Testing data set 7.

The features for object 7 are:- cmp= 37, mnx= 46, mnd= 8, mdx= 40.
Element in Look-up table is 24.

This object corresponds to file 0, cluster 4.
The description of this file is "cup8*7__."
The number of test images in the object cluster this belongs to is:- 28.

These points correspond to angles of orientation from 40 to 66 degrees.
The distances of this object's centre from the three end points are: 30, 30, 30.

Testing data set 9.
Compaction feature for data set 9 is out of range.

Testing data set 21.

The features for object 21 are:- cmp= 105, mnx= 94, mnd= 42, mdx= 74.
Element in Look-up table is 43.

This object corresponds to file 1, cluster 3.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 63.

These points correspond to angles of orientation from 16 to 72 degrees.
The distances of this object's centre from the three end points are: 48, 46, 43.

Testing data set 25.

The features for object 25 are:- cmp= 41, mnx= 59, mnd= 26, mdx= 38.
Element in Look-up table is cff.

A cluster collision has occured at this point.
The collision index of this collision is 12. The clusters colliding are:-

file 0, cluster 10.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 127.
These points correspond to angles of orientation from 1 to 360 degrees.
The distances of the pick-up point from the three end points are: 31, 28, 30.
file 1, cluster 0.
The file description is bowl8*7_.
The number of test images in the object cluster this belongs to is:- 11.
These points correspond to angles of orientation from 81 to 90 degrees.
The distances of the pick-up point from the three end points are: 45, 44, 45.


Testing data set 32.

The features for object 32 are:- cmp= 34, mnx= 52, mnd= 30, mdx= 31.
Element in Look-up table is 3ff.

A cluster collision has occured at this point.
The collision index of this collision is 3. The clusters colliding are:-

file 0, cluster 5.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 133 to 356 degrees.
The distances of the pick-up point from the three end points are: 31, 29, 31.

file 0, cluster 9.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 22 to 323 degrees.
The distances of the pick-up point from the three end points are: 31, 30, 29.
```

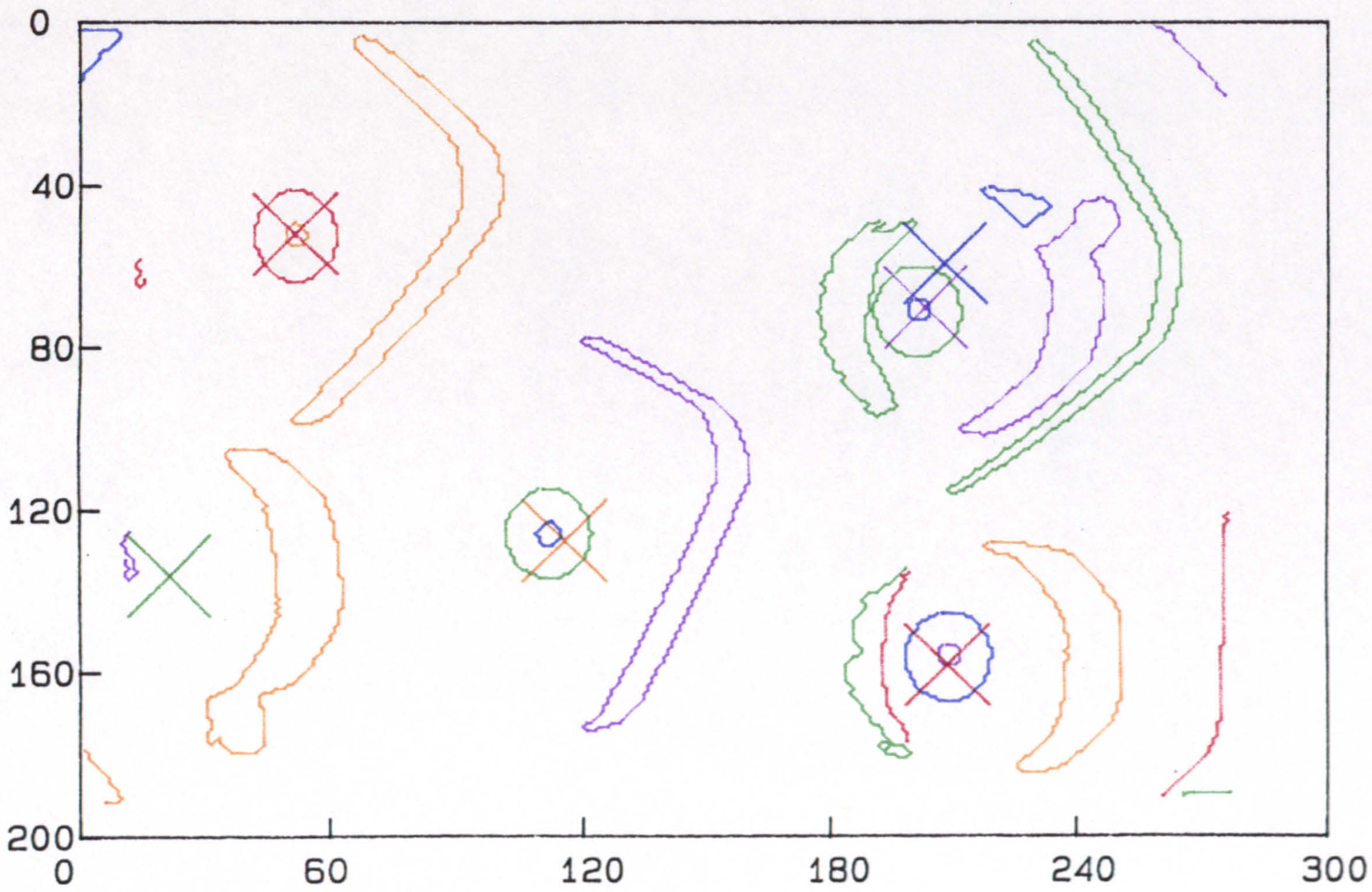
TABLE of IMAGE FEATURES

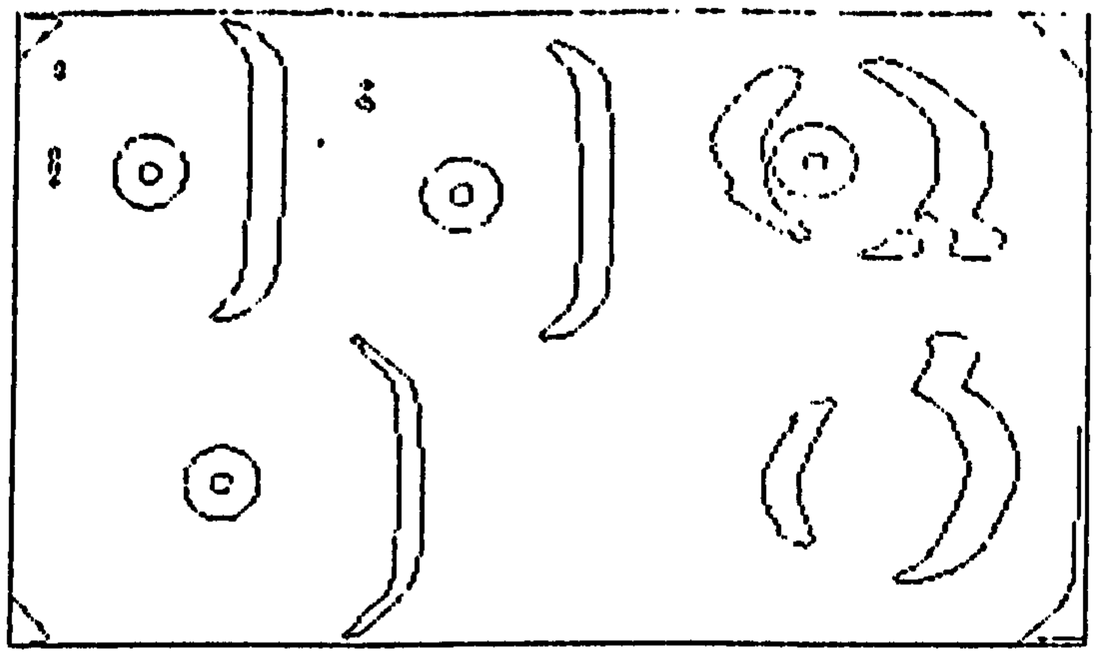
Figure 12.39a Continued.

Sample	minpoint i	j	pid	midpoint i	j	pid	maxpoint i	j	pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
2	67	5	3	92	35	33	53	95	93	203	254	746	86	39	71	91	110
3	231	5	1	263	55	51	212	114	121	253	311	654	147	59	77	110	136
5	0	0	0	0	0	0	0	0	0	58	69	389	12	0	0	0	0
7	232	56	19	236	64	27	215	99	65	143	172	795	37	8	40	46	48
9	178	71	27	214	70	114	200	82	133	182	219	204	235	36	18	24	54
21	120	78	1	153	104	39	122	172	107	229	271	695	105	42	74	94	116
25	35	106	2	47	130	26	32	165	63	179	204	1013	41	26	38	59	64
26	0	0	0	0	0	0	0	0	0	59	70	463	10	0	0	0	0
27	0	0	0	0	0	0	0	0	0	71	0	0	0	0	0	0	0
32	219	129	1	240	151	28	229	181	59	148	170	826	34	30	31	52	61
33	0	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	61	71	503	10	0	0	0	0

PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.39b





Print of current image being worked on.

Figure 12.40a

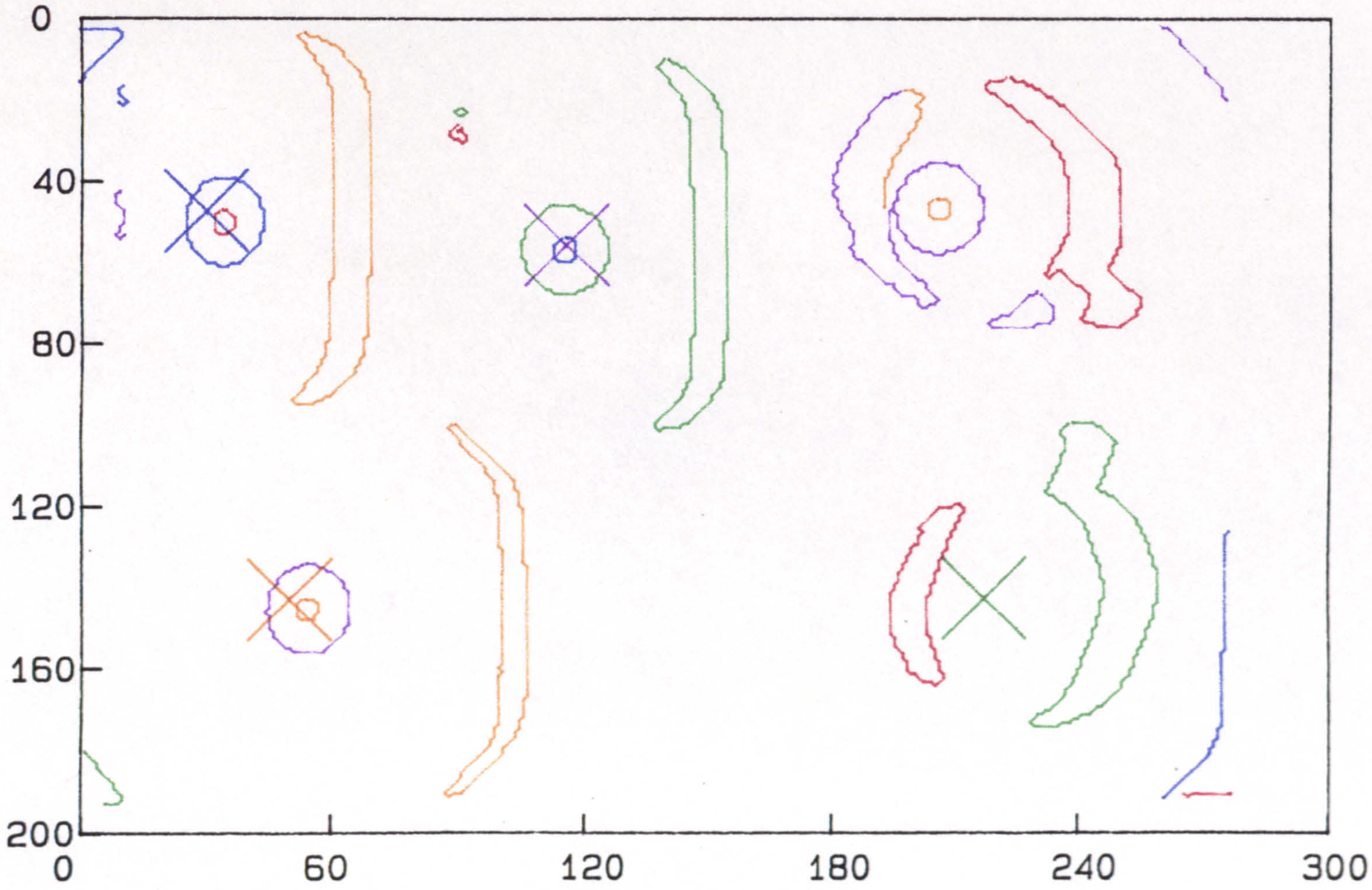
```
Object identification by mapping into the look-up table.
Testing data set 2.
The features for object 2 are:- cmp= 60, mnx= 88, mnd= 15, mdx= 75.
Element in Look-up table is 47.

This object corresponds to file 1, cluster 7.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 15.

These points correspond to angles of orientation from 1 to 14 degrees.
The distances of this object's centre from the three end points are: 48, 44, 48.

Testing data set 4.

The features for object 4 are:- cmp= 58, mnx= 86, mnd= 31, mdx= 56.
Element in Look-up table is 42.

This object corresponds to file 1, cluster 2.
The description of this file is "bowl8*7_."
The number of test images in the object cluster this belongs to is:- 4.

These points correspond to angles of orientation from 0 to 180 degrees.
The distances of this object's centre from the three end points are: 50, 35, 47.

Testing data set 5.

The features for object 5 are:- cmp= 39, mnx= 45, mnd= 29, mdx= 21.
Element in Look-up table is 0.

No match with learnt objects found for this object.

Testing data set 23.

The features for object 23 are:- cmp= 98, mnx= 88, mnd= 32, mdx= 59.
Element in Look-up table is 62.

This object corresponds to file 2, cluster 2.
The description of this file is "plate8*7."
The number of test images in the object cluster this belongs to is:- 22:.

These points correspond to angles of orientation from 87 to 106 degrees.
The distances of this object's centre from the three end points are: 54, 55, 56.

Testing data set 24.

The features for object 24 are:- cmp= 34, mnx= 55, mnd= 26, mdx= 37.
Element in Look-up table is 2ff.

A cluster collision has occured at this point.
The collision index of this collision is 2. The clusters colliding are:-

file 0, cluster 6.
The file description is cup8*7__.
The number of test images in the object cluster this belongs to is:- 44.
These points correspond to angles of orientation from 24 to 275 degrees.
The distances of the pick-up point from the three end points are: 31, 29, 32.

file 0, cluster 9.
The file description is cup8*7  .
The number of test images in the object cluster this belongs to is:- 37.
These points correspond to angles of orientation from 22 to 323 degrees.
The distances of the pick-up point from the three end points are: 31, 30, 29.
```

TABLE of IMAGE FEATURES

Sample	minpoint i	minpoint j	minpoint pid	midpoint i	midpoint j	midpoint pid	maxpoint i	maxpoint j	maxpoint pid	Bound	Perim	Area	Compact	d(mn-md)	d(mx-md)	d(mn-mx)	Total
2	53	2	1	61	15	14	53	90	89	195	208	721	60	15	75	88	90
4	139	10	2	148	49	32	140	96	88	192	204	713	58	31	56	86	87
5	220	16	6	240	38	31	236	59	52	153	183	845	39	29	21	45	50
8	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	58	67	367	12	0	0	0	0
16	0	0	0	0	0	0	0	0	0	60	70	403	12	0	0	0	0
23	88	99	1	101	129	31	89	187	89	187	209	443	98	32	59	88	91
24	234	115	17	249	137	41	232	170	76	175	179	932	34	26	37	55	63
26	196	150	36	196	150	36	196	150	36	98	113	481	26	0	0	0	0
27	0	0	0	0	0	0	0	0	0	67	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	59	69	472	10	0	0	0	0

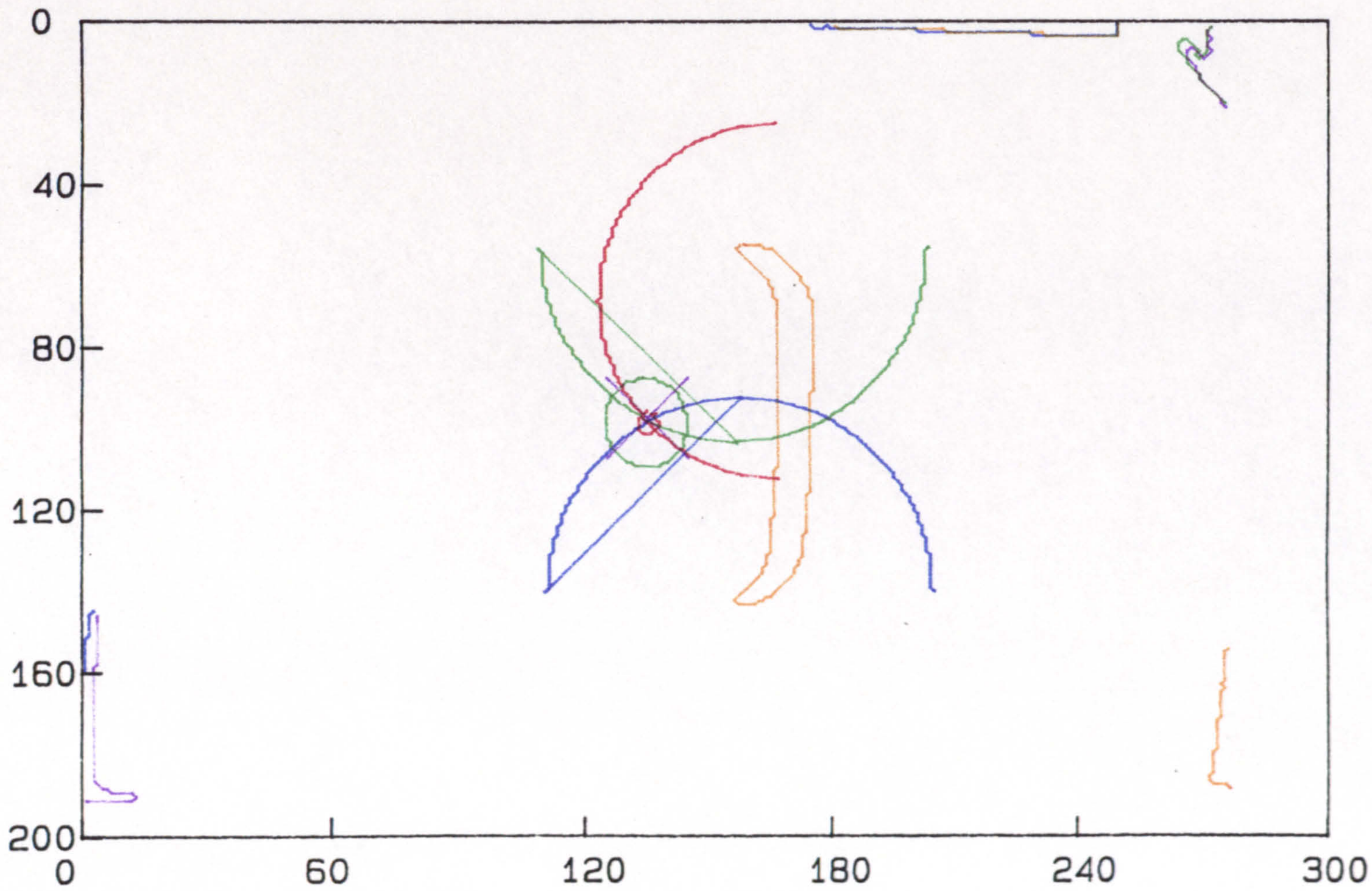
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.40b



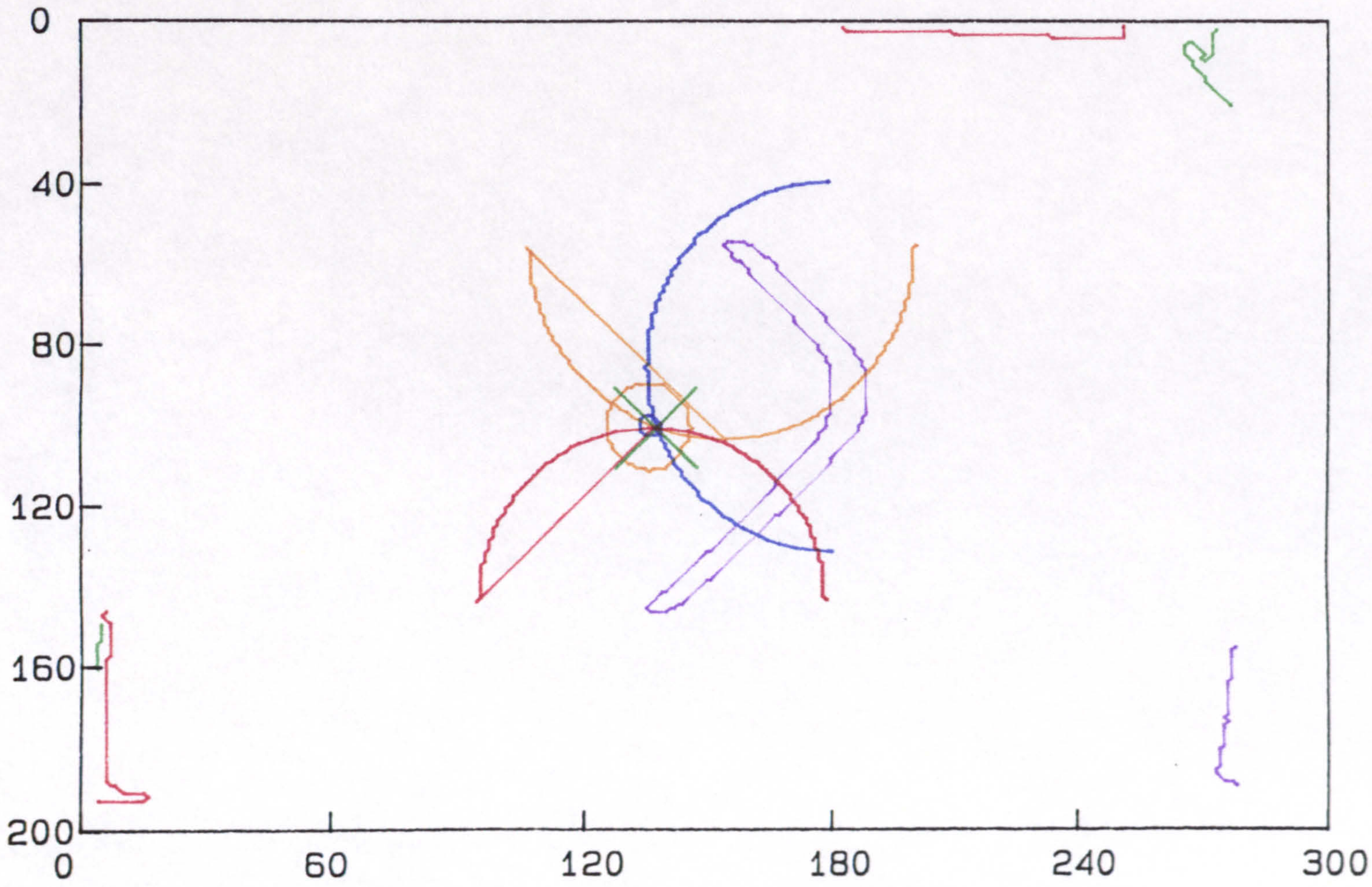
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.41



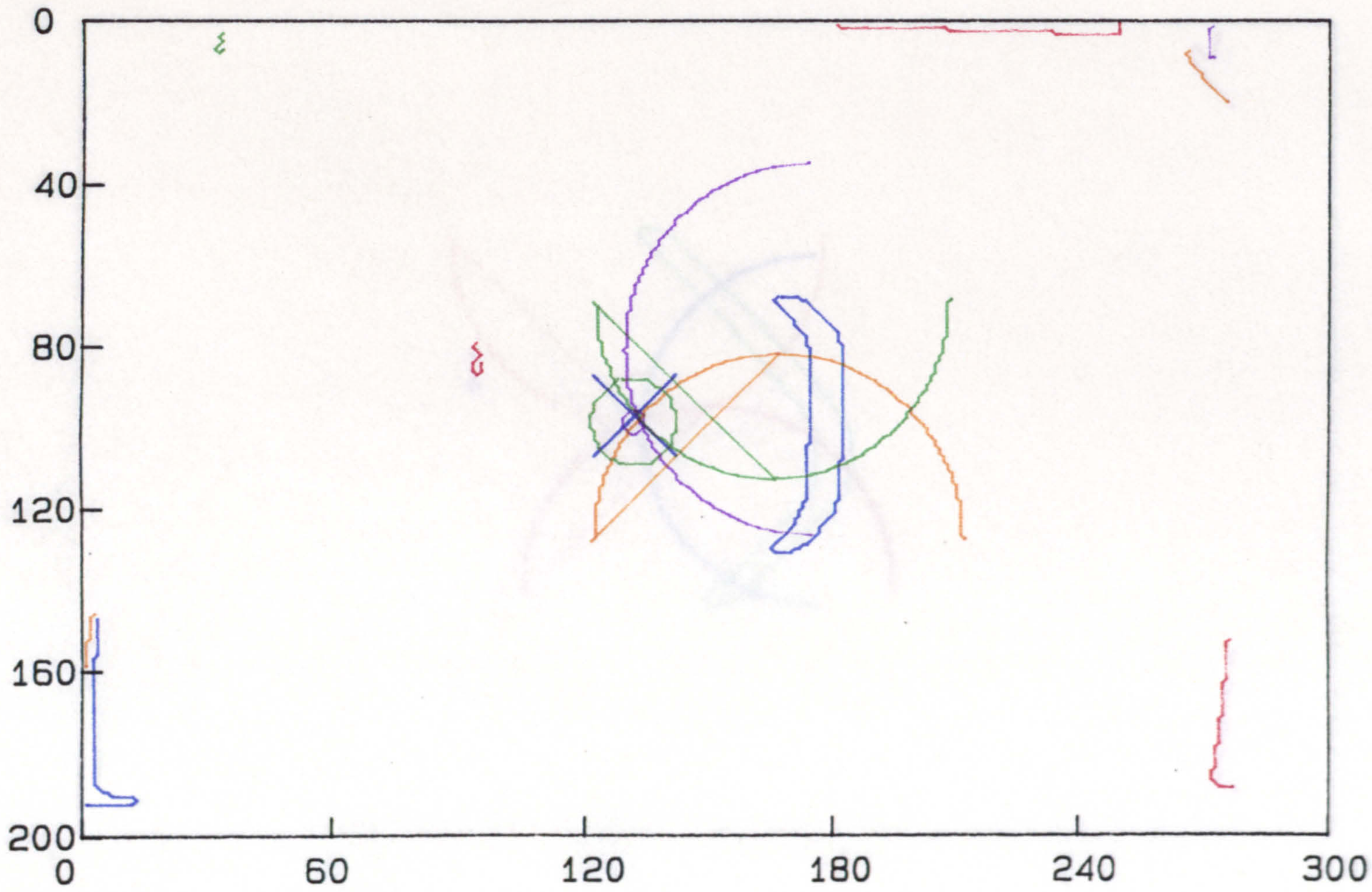
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.42



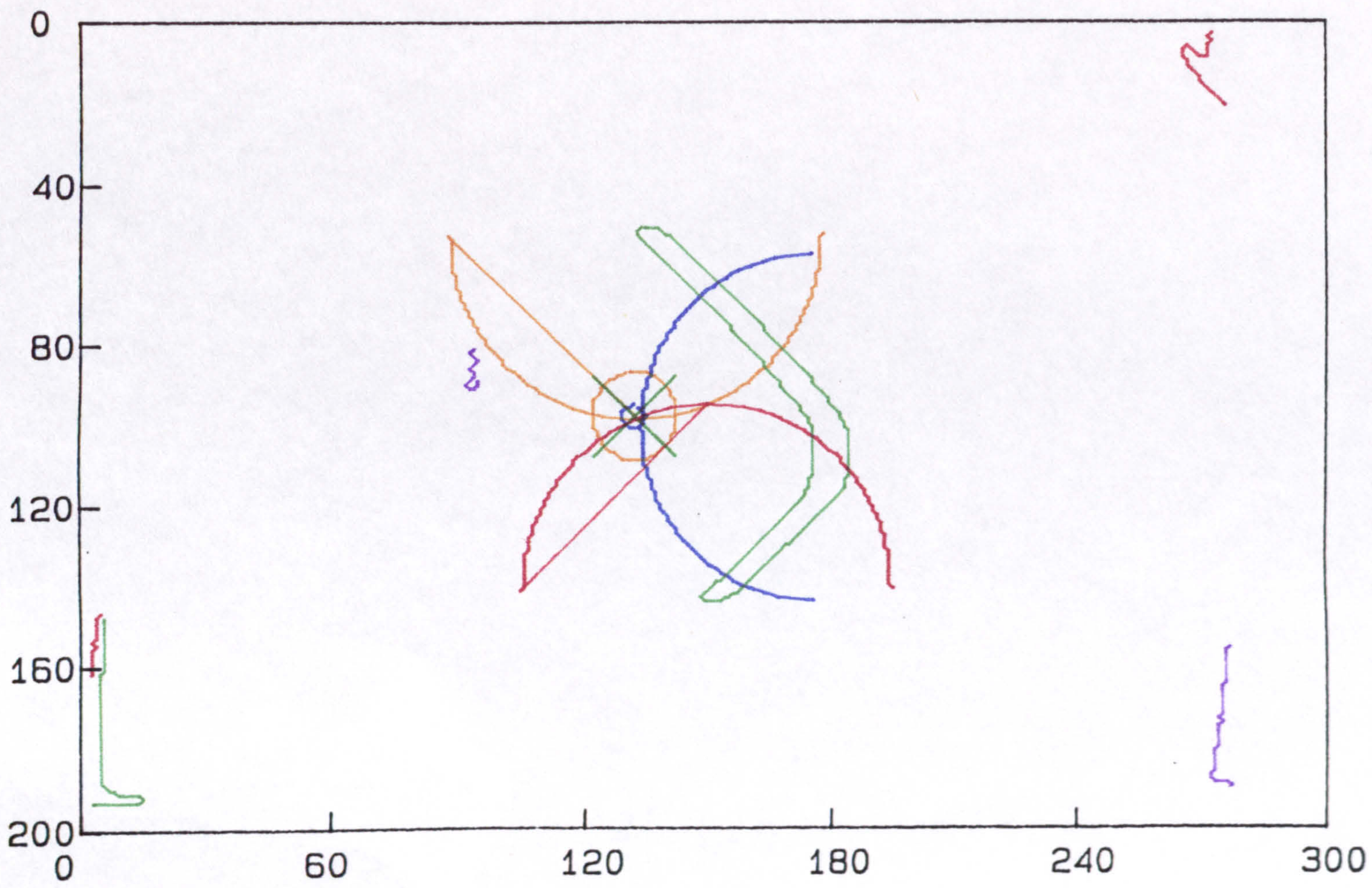
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.43



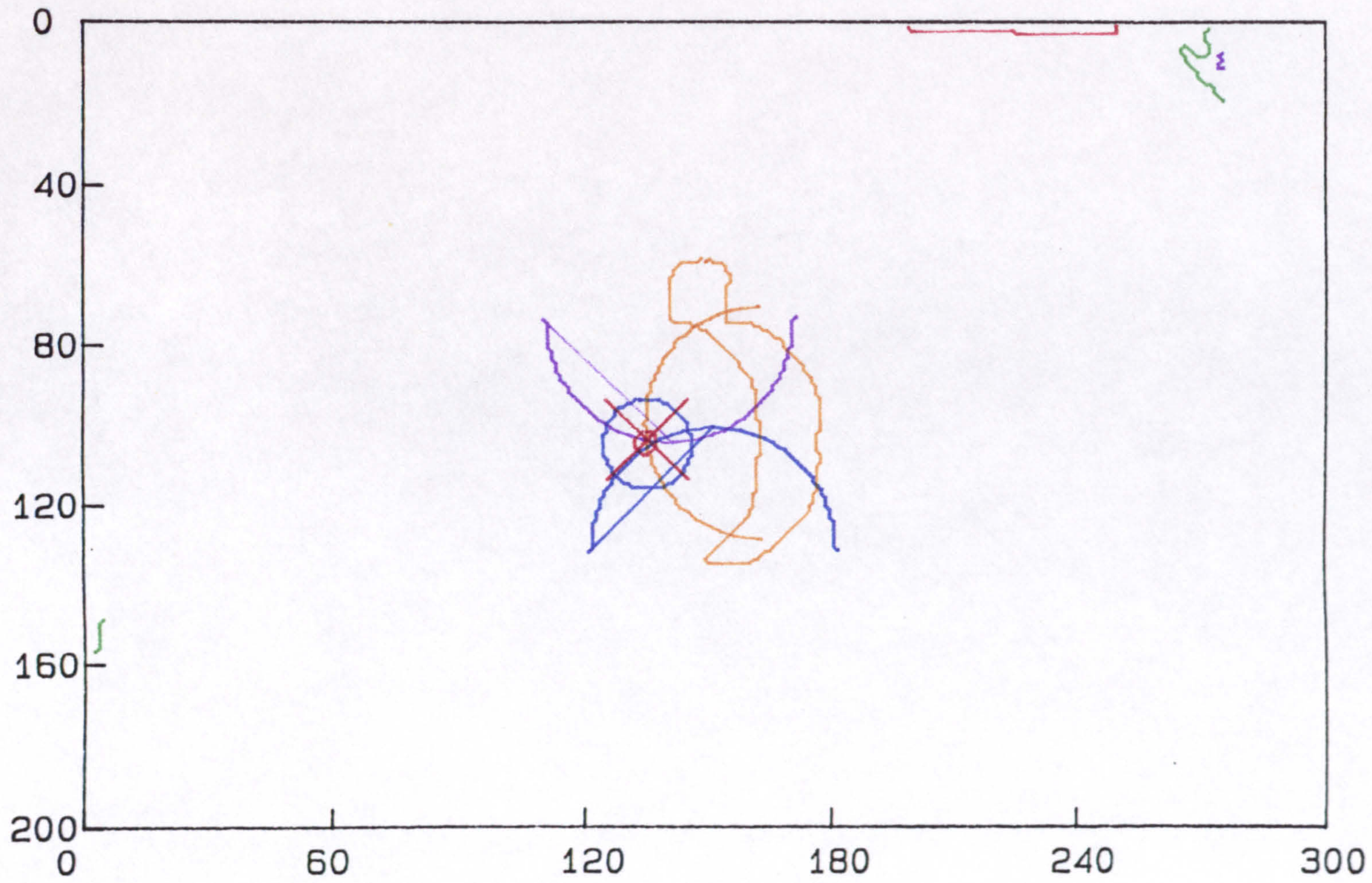
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.44



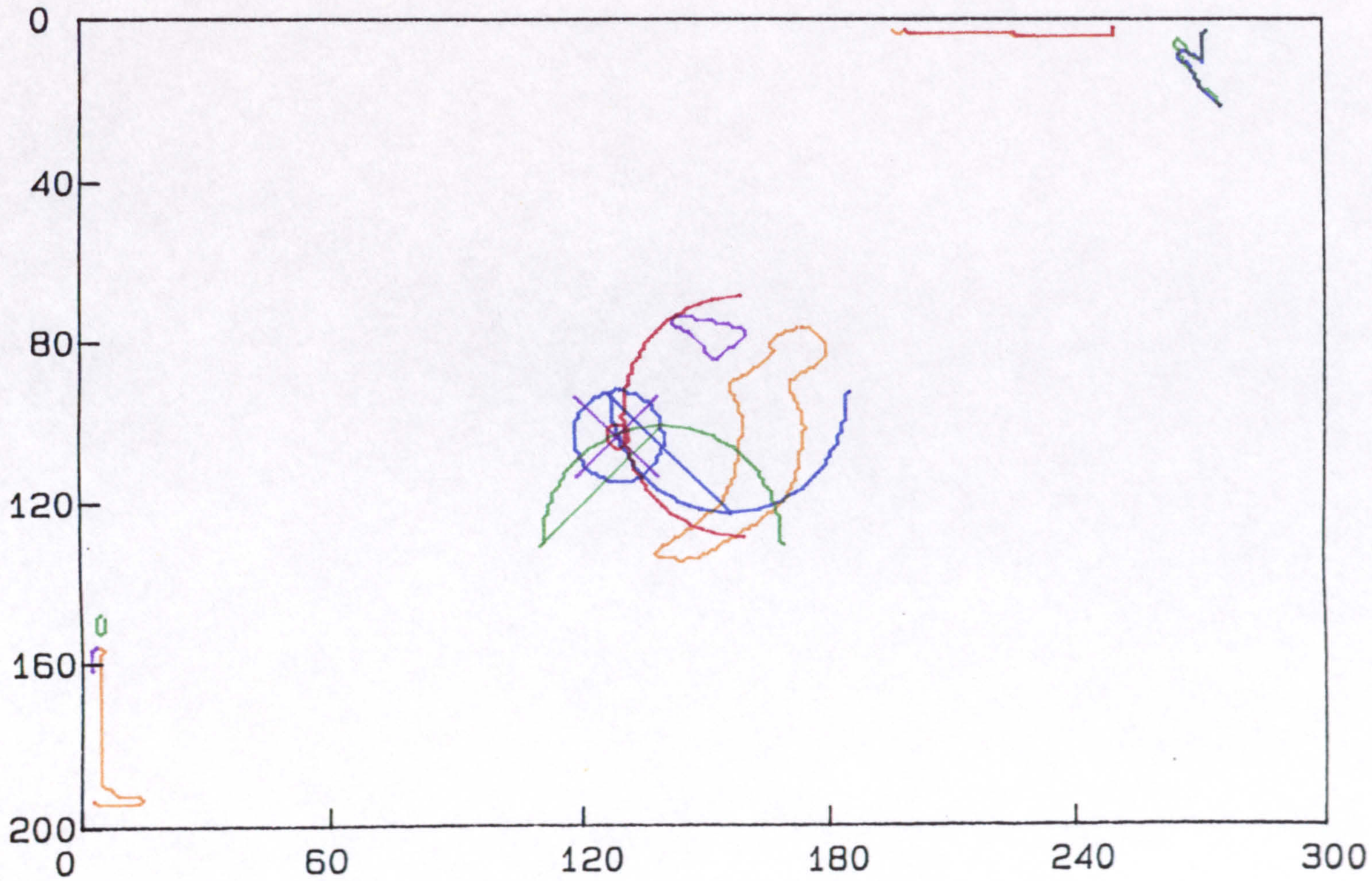
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.45



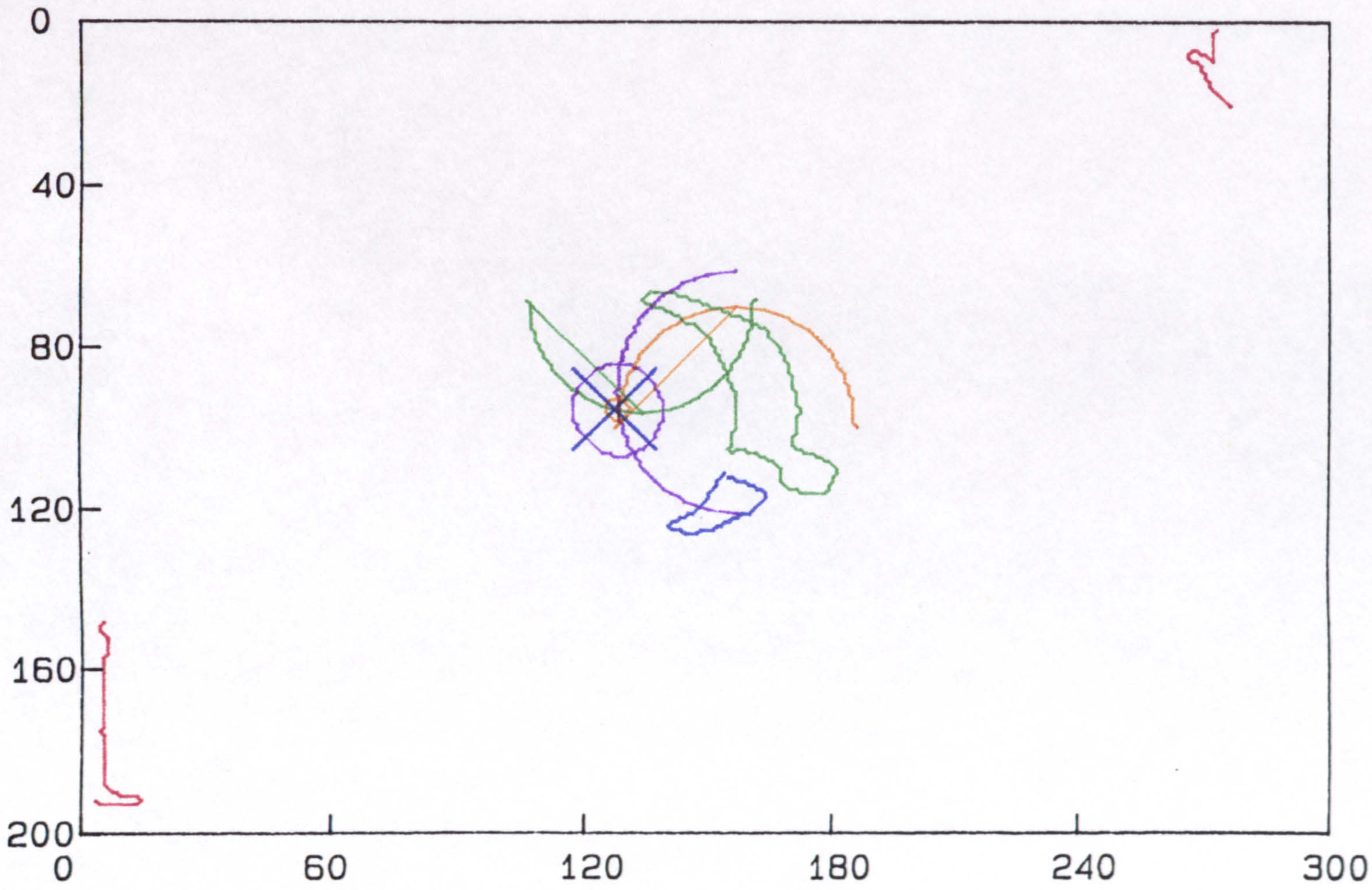
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.46



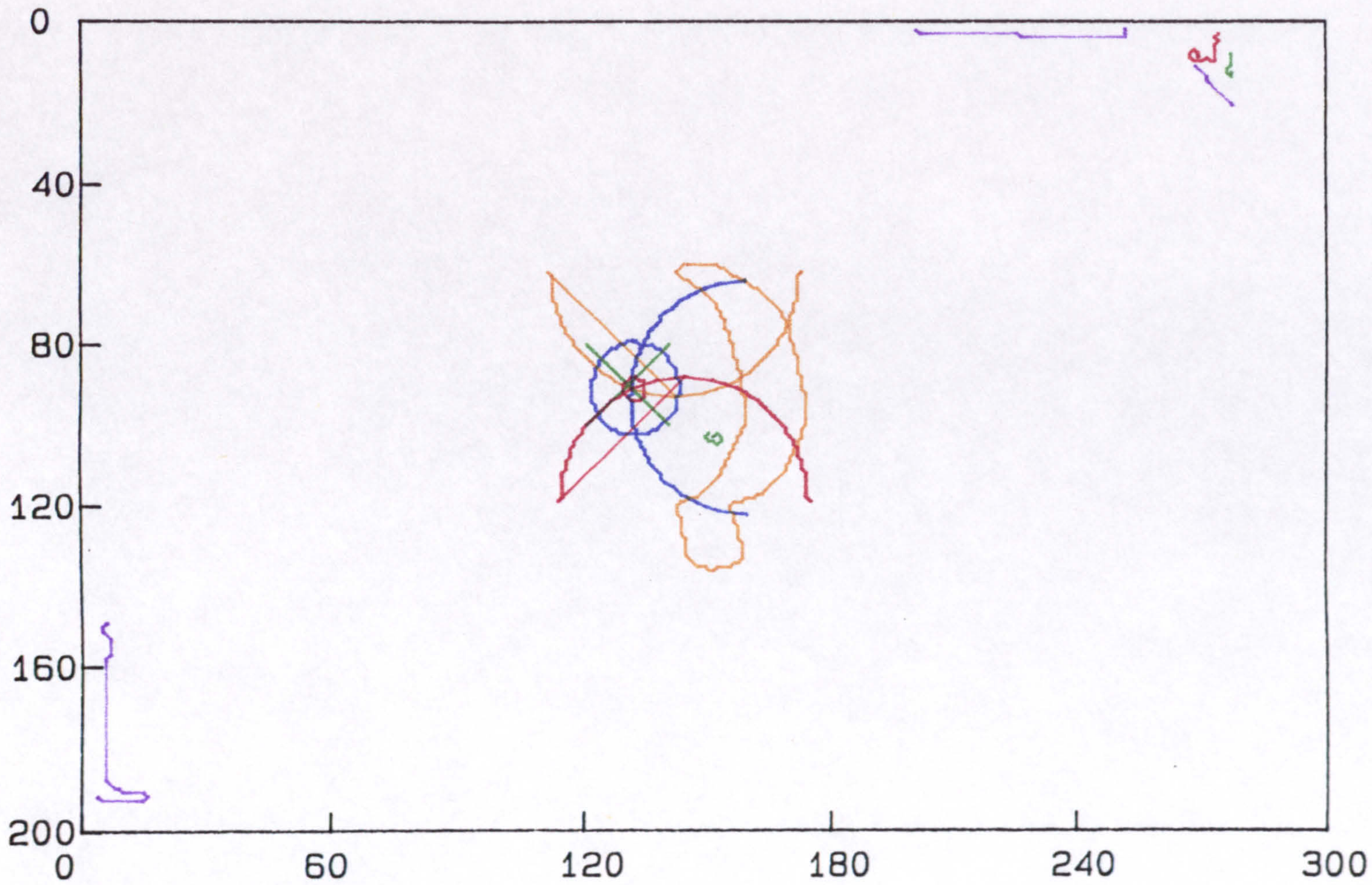
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.47



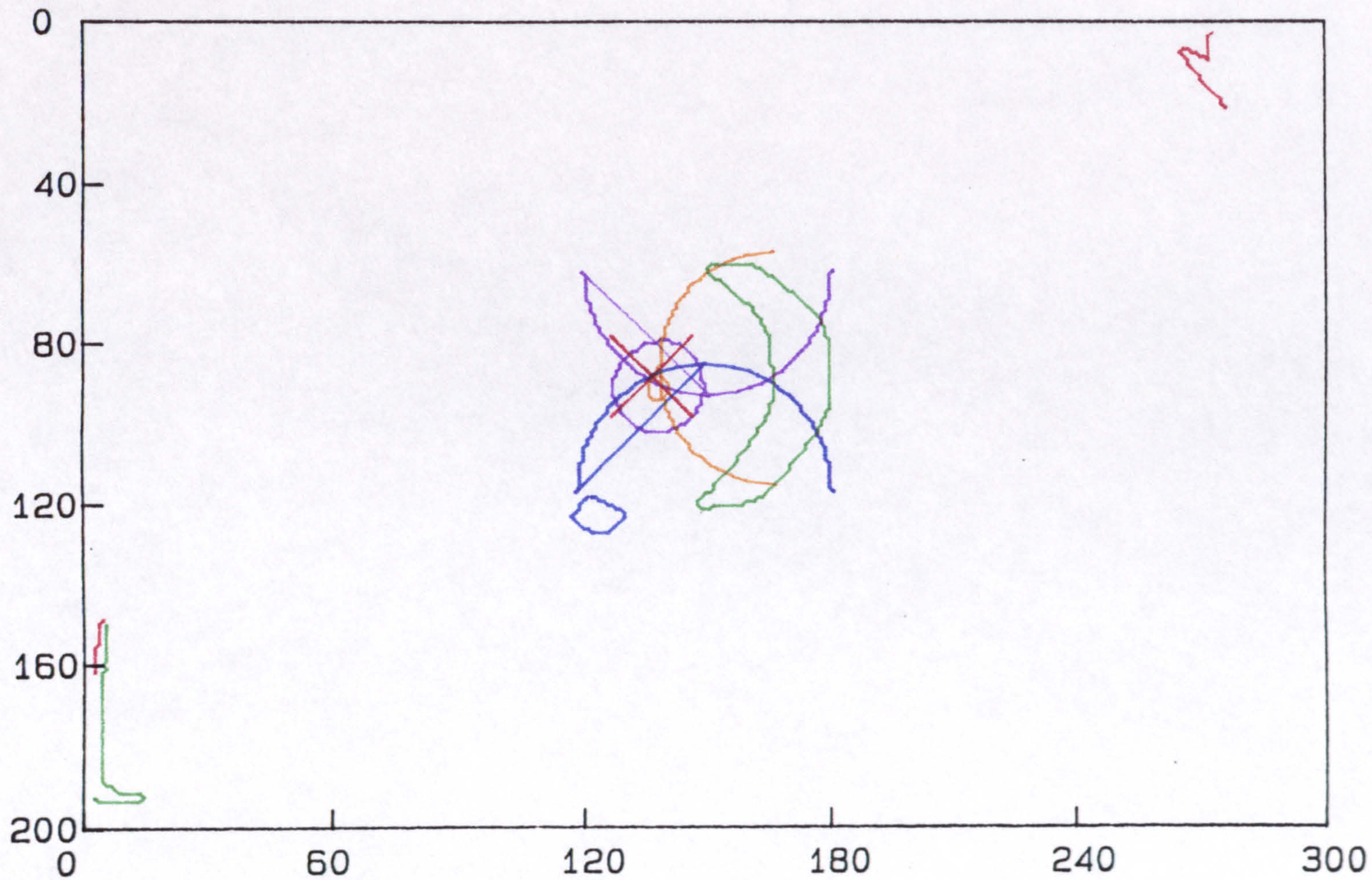
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.48



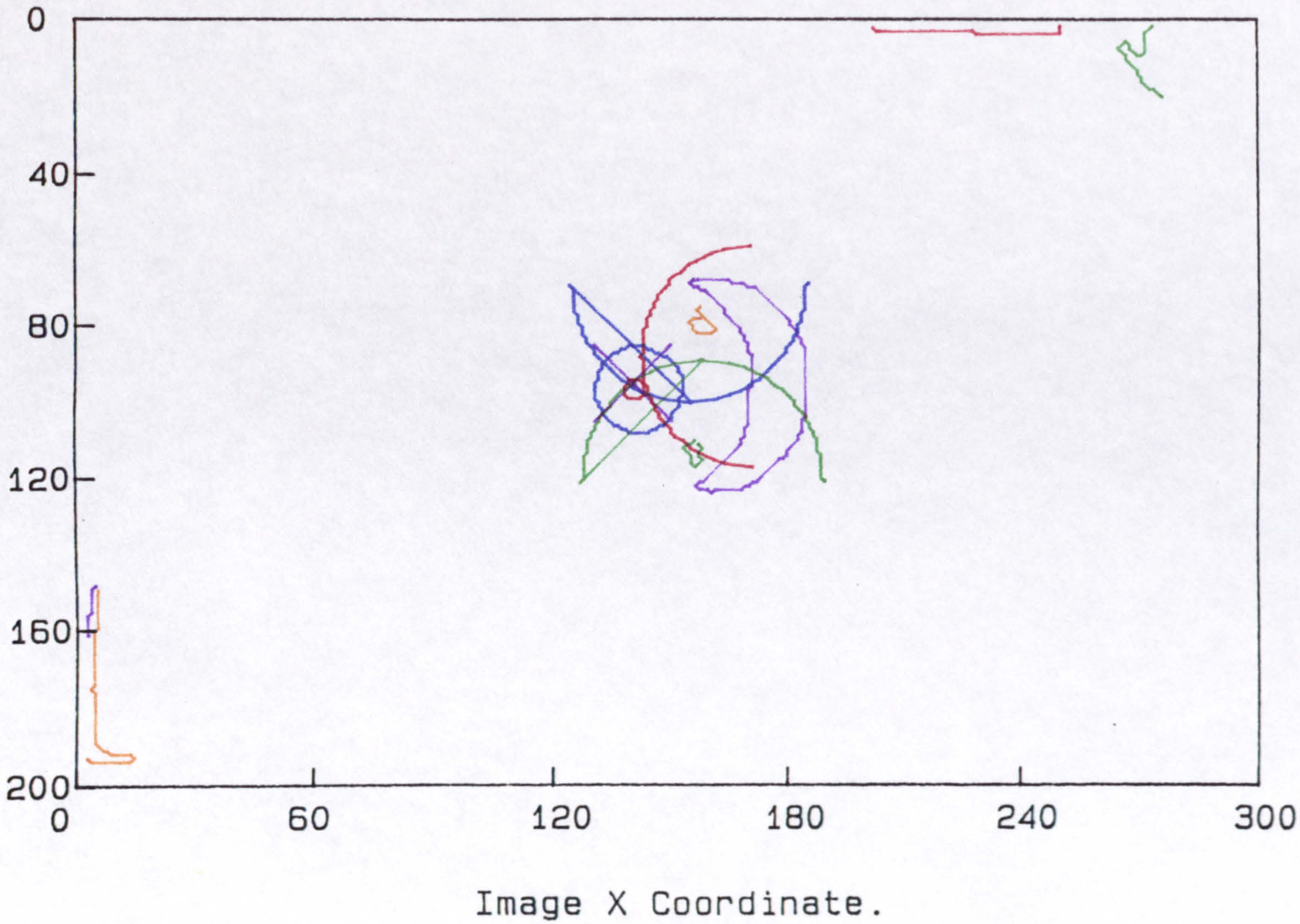
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.49



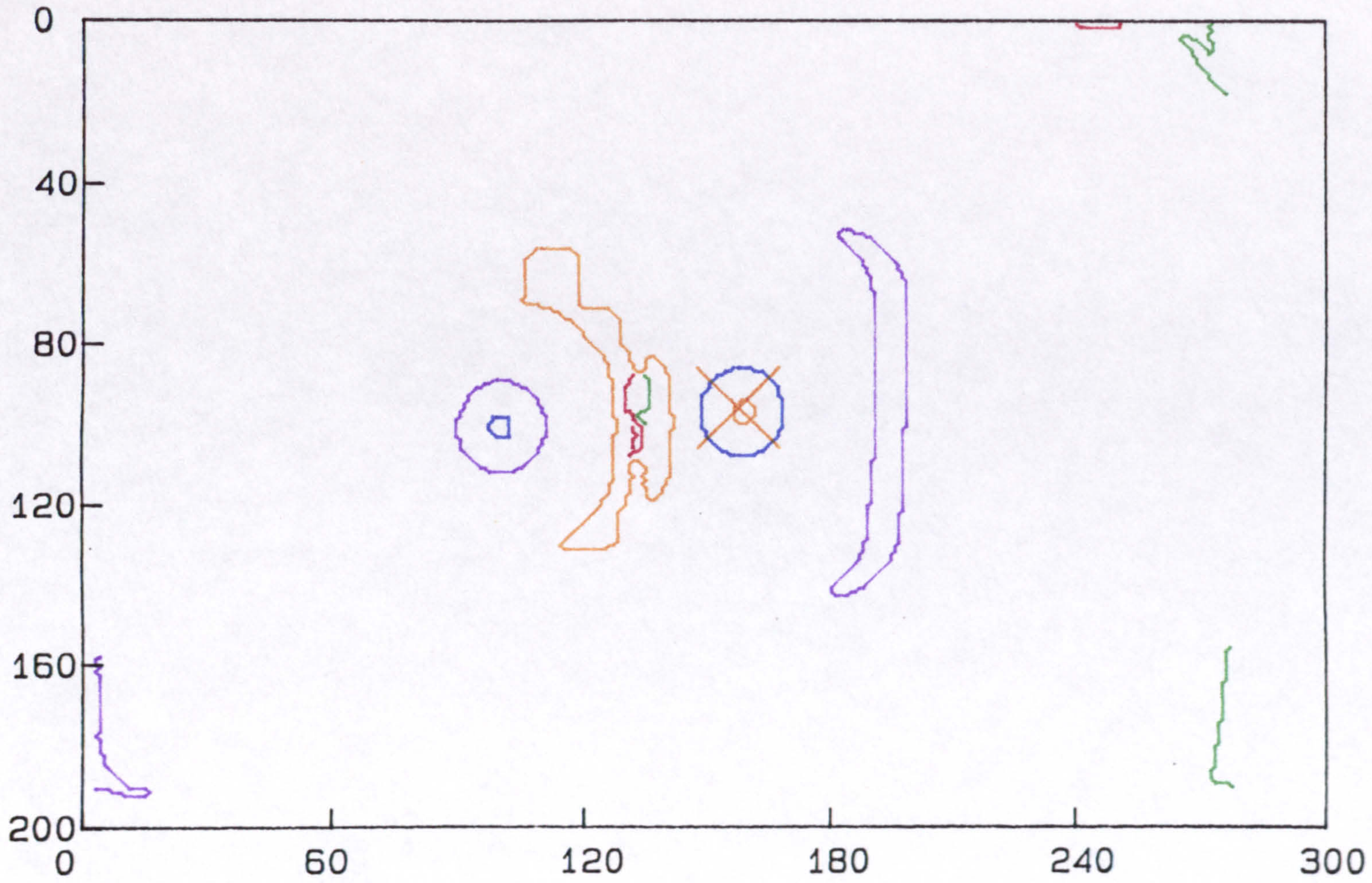
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.50



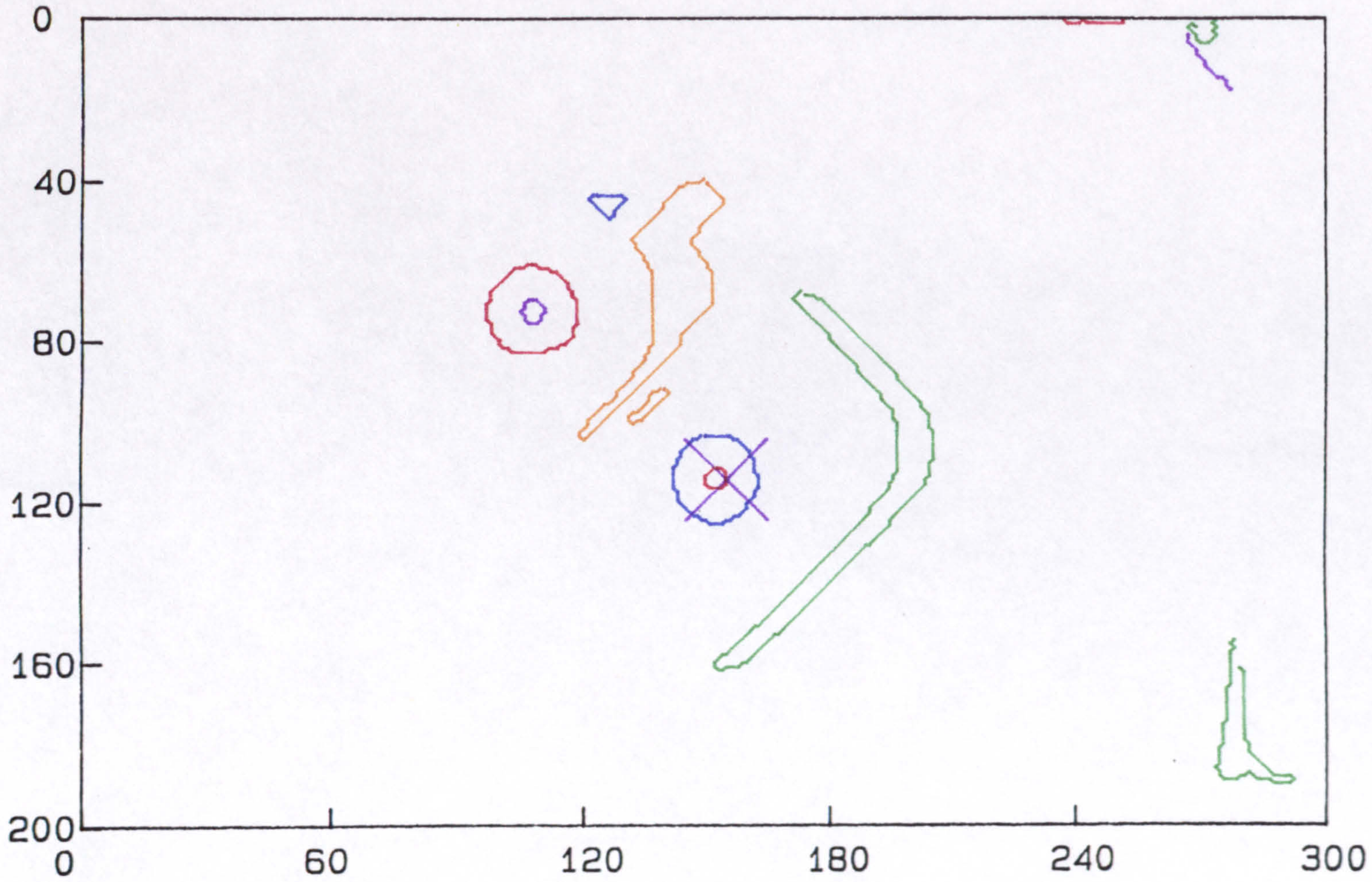
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.51



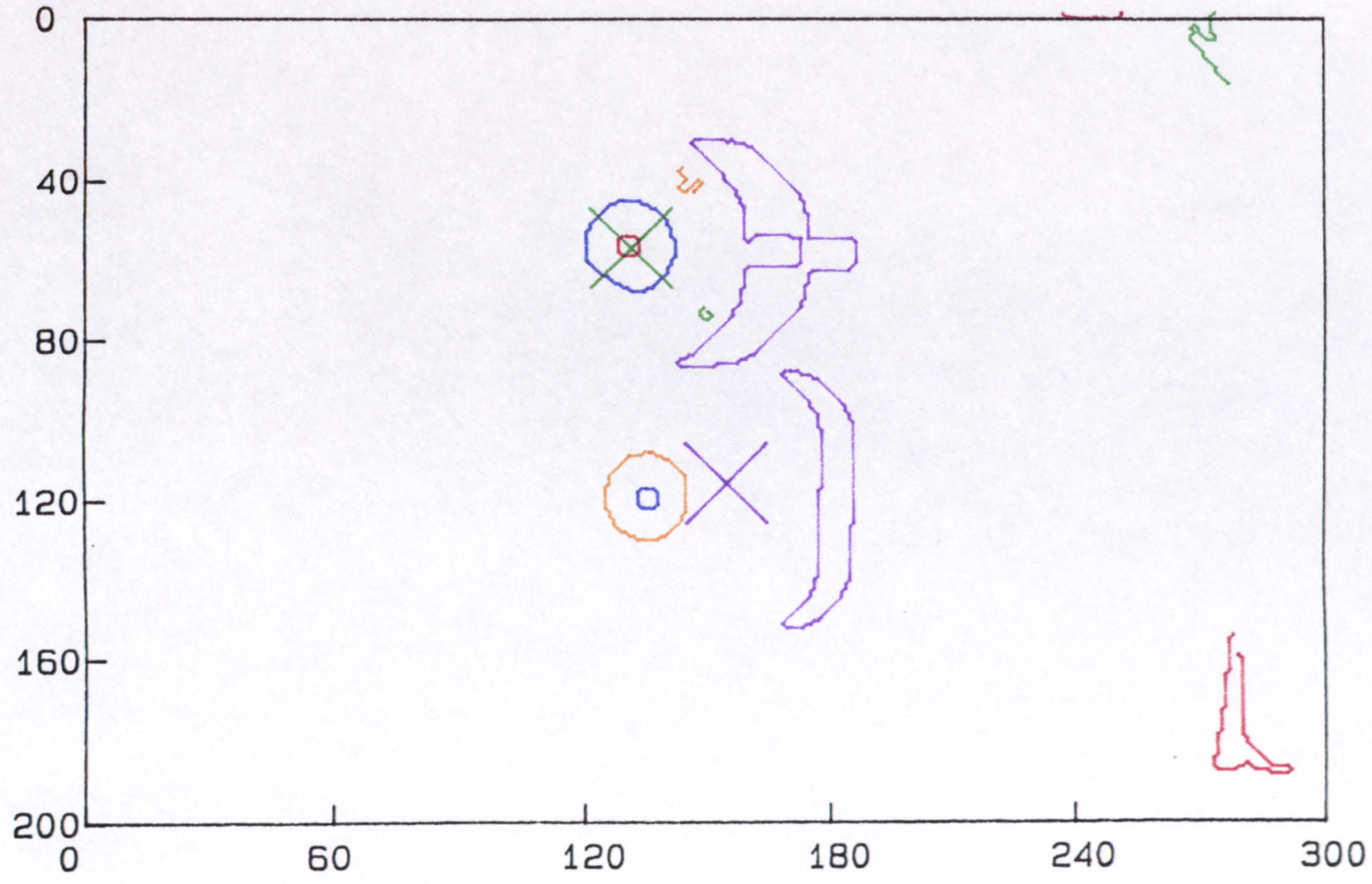
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.52



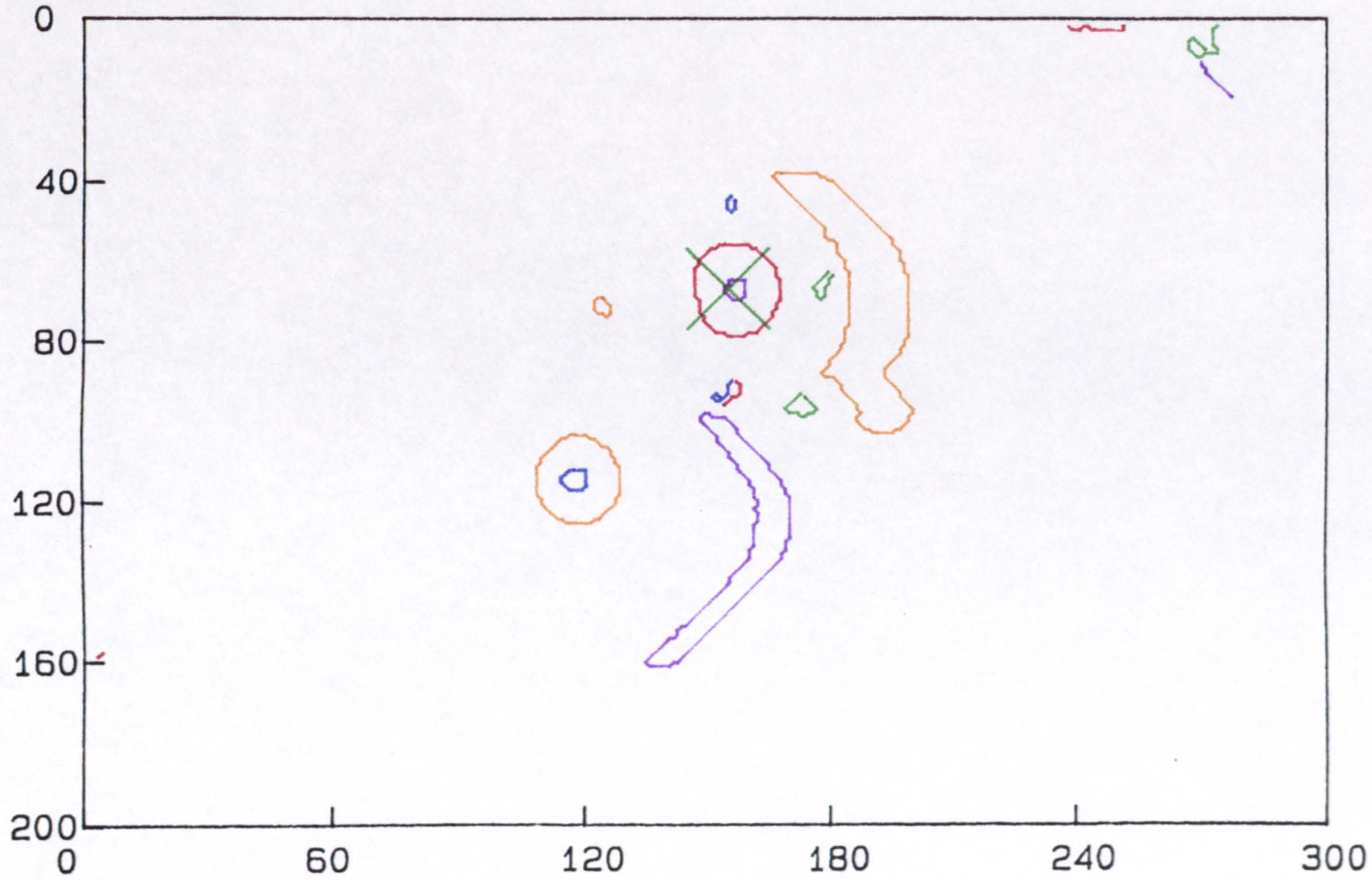
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.53



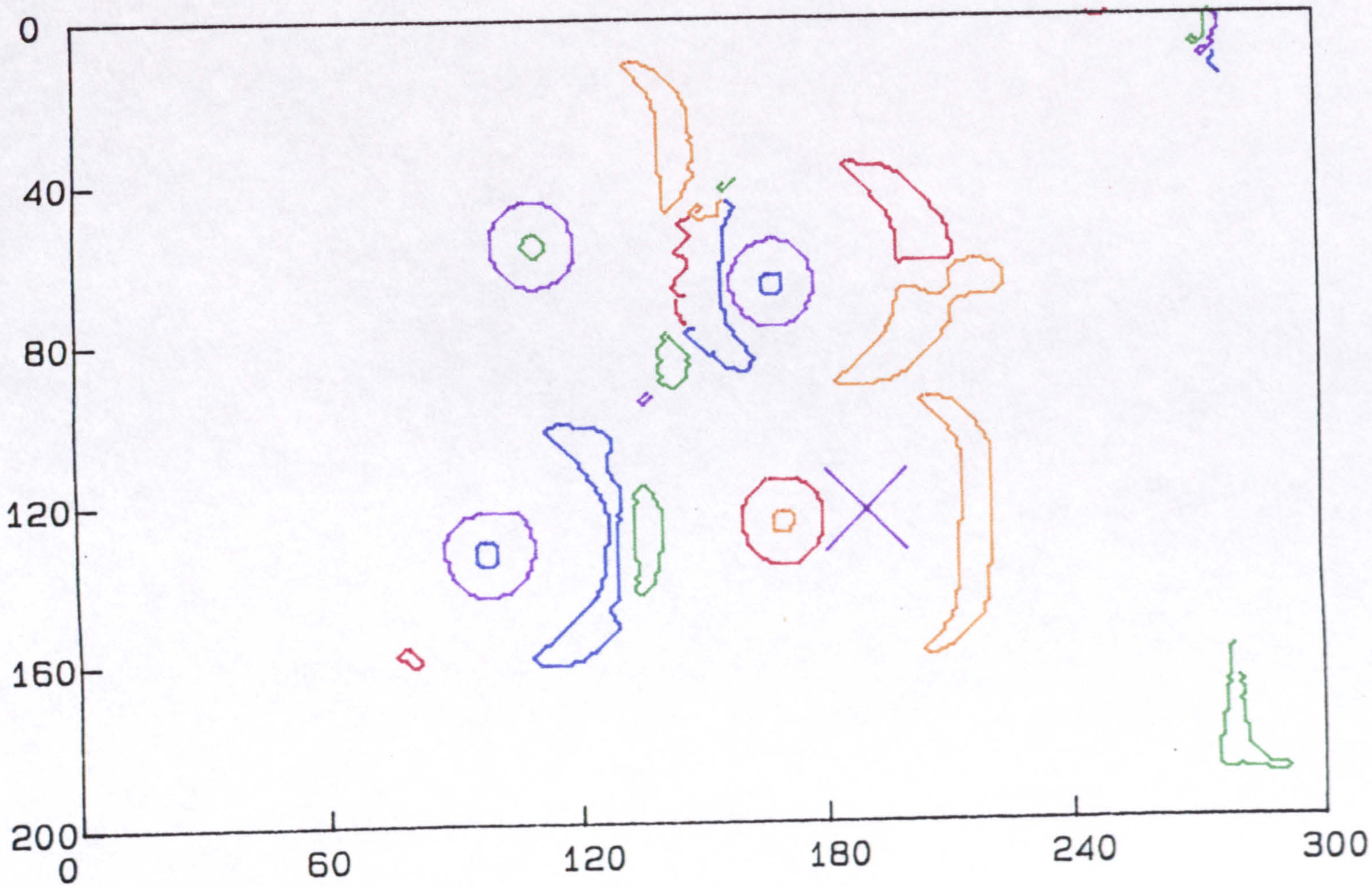
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.54



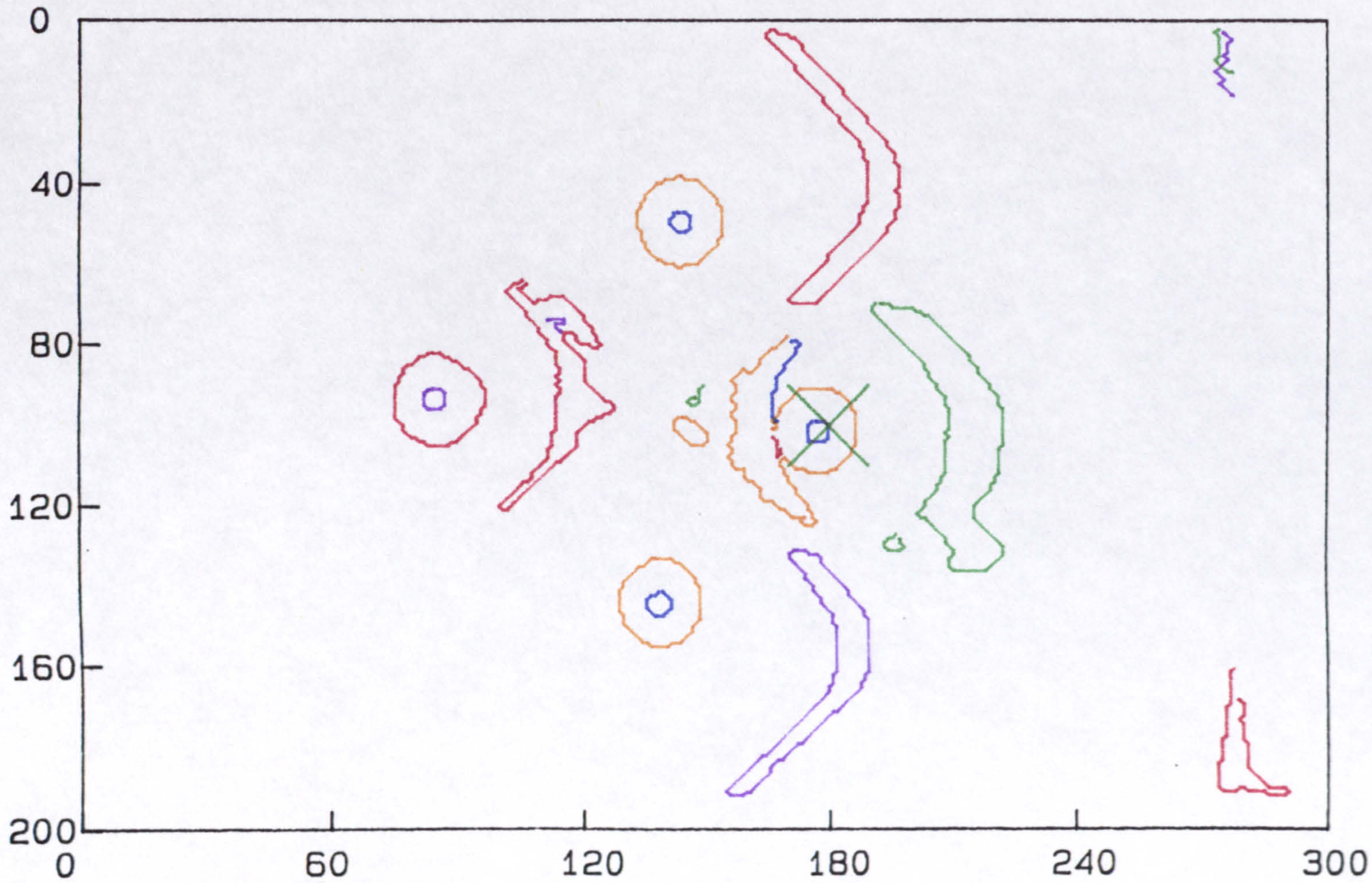
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.55



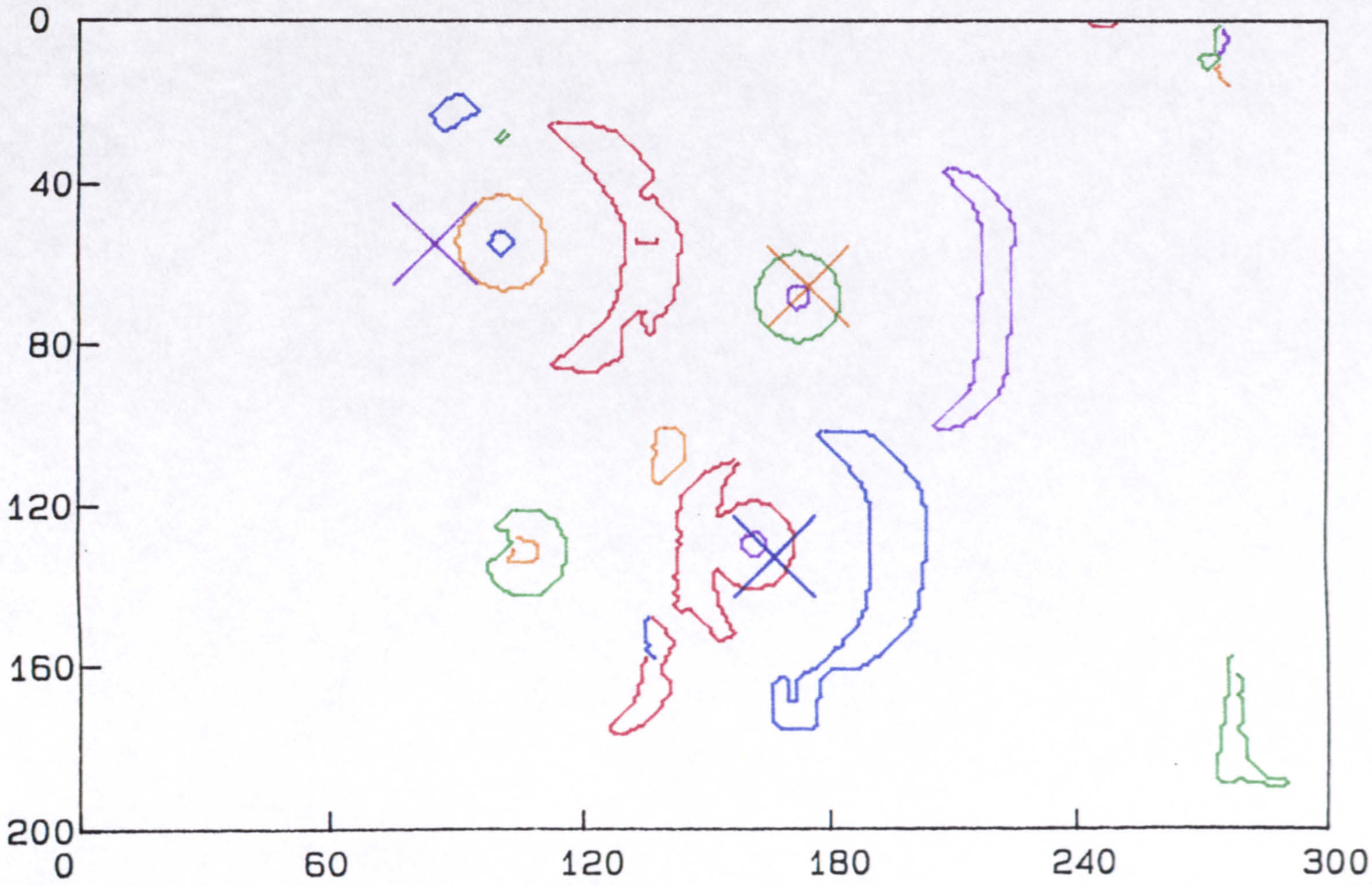
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.56



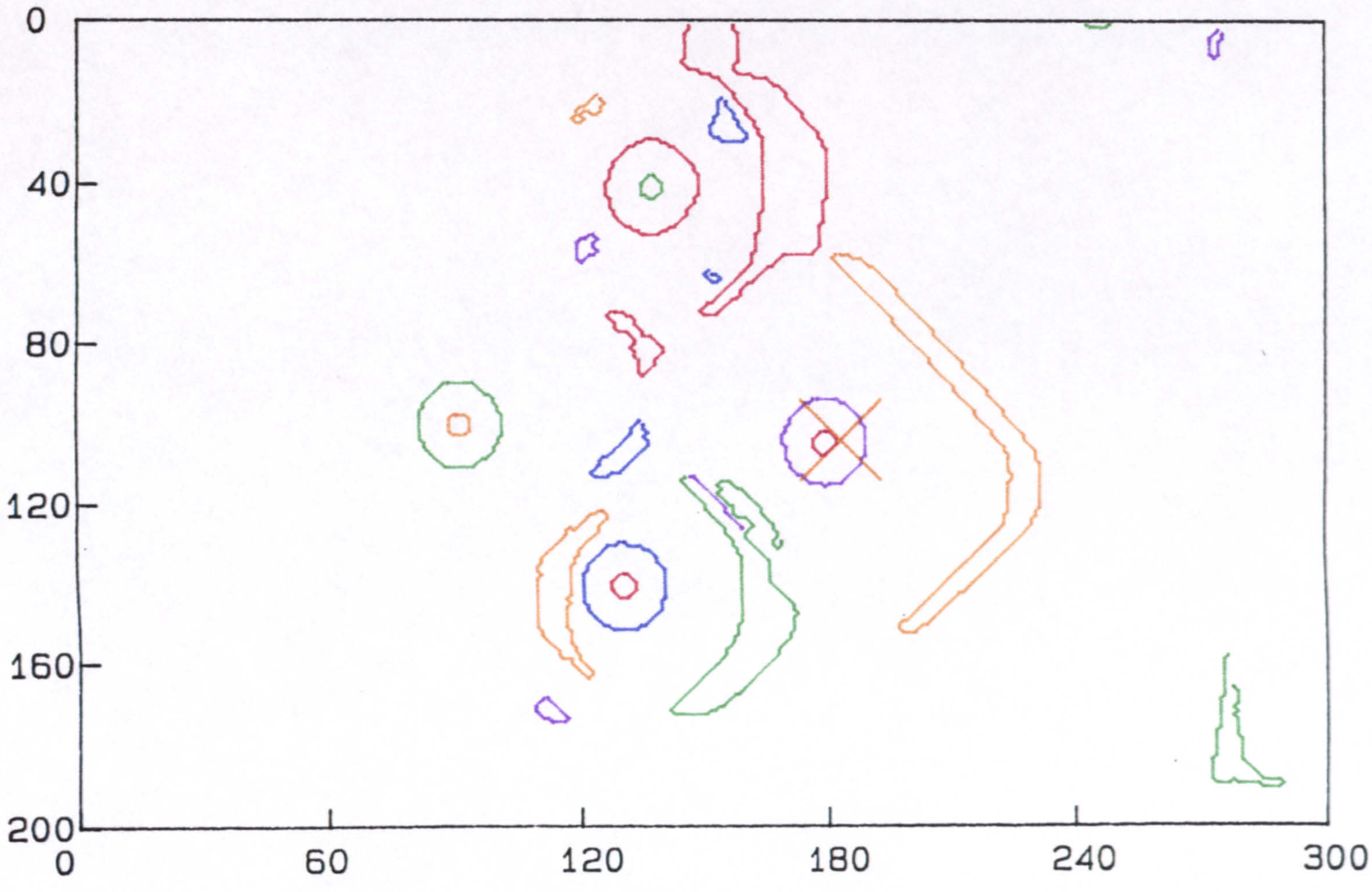
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.57



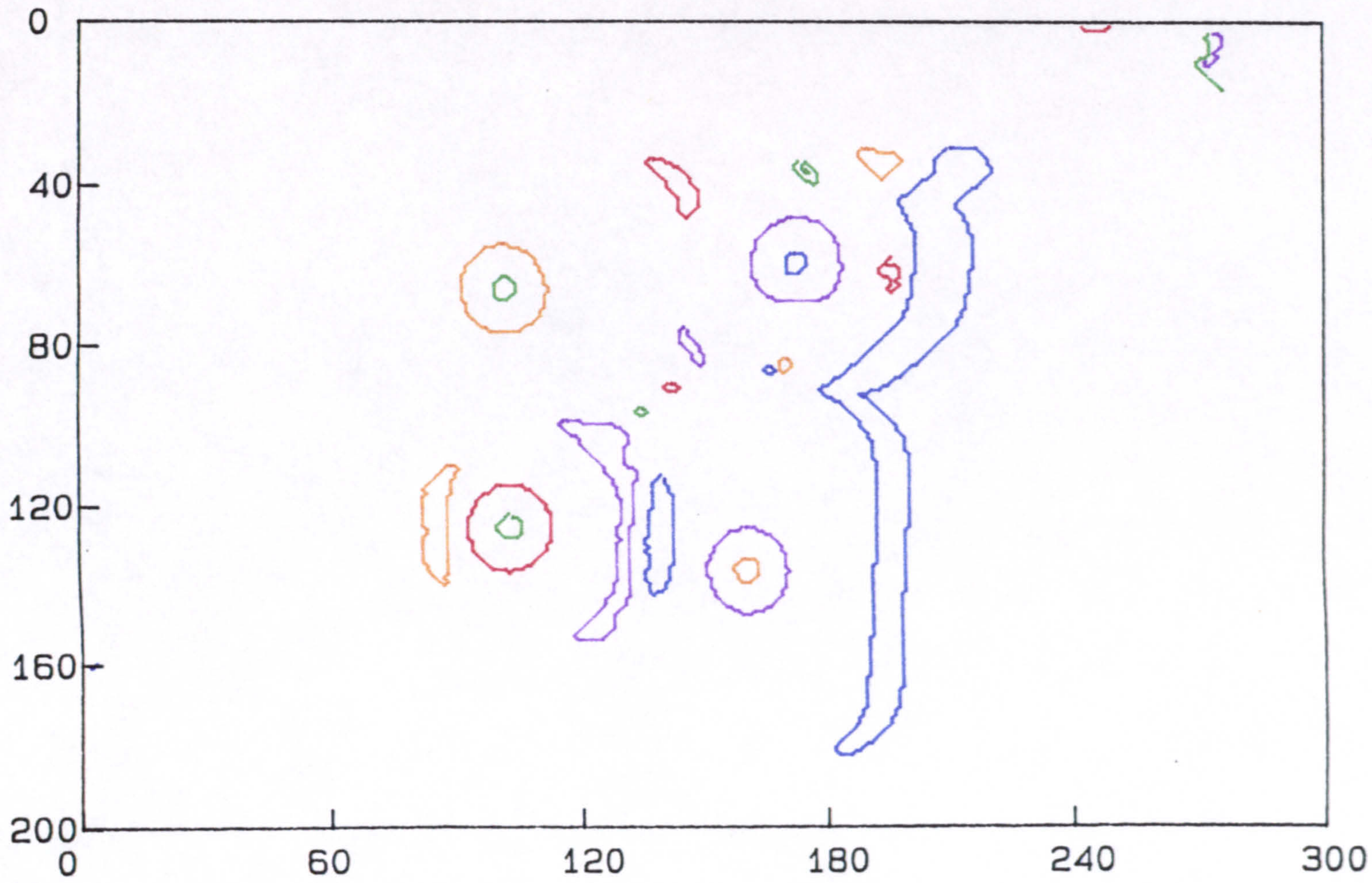
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.58



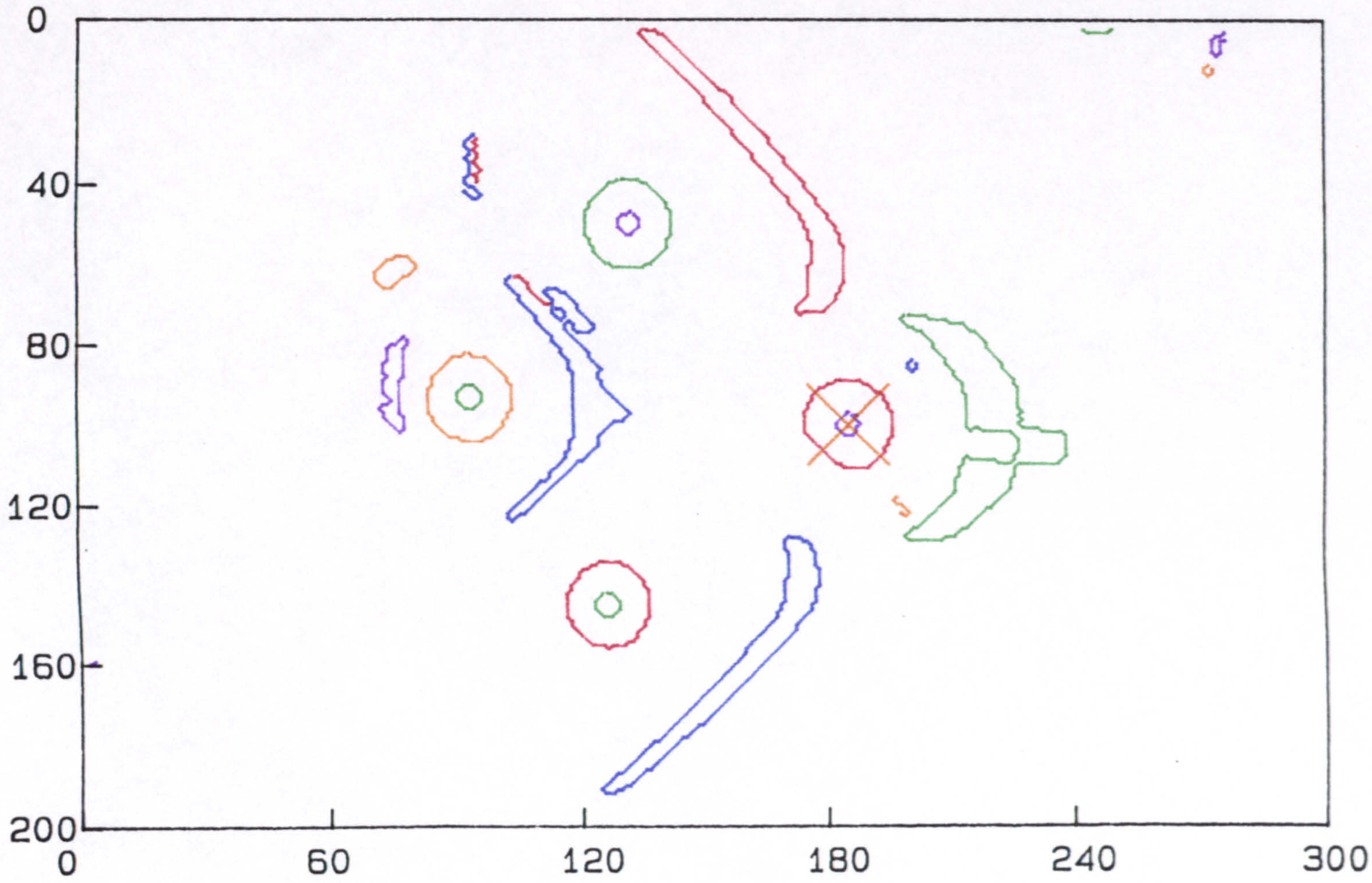
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.59



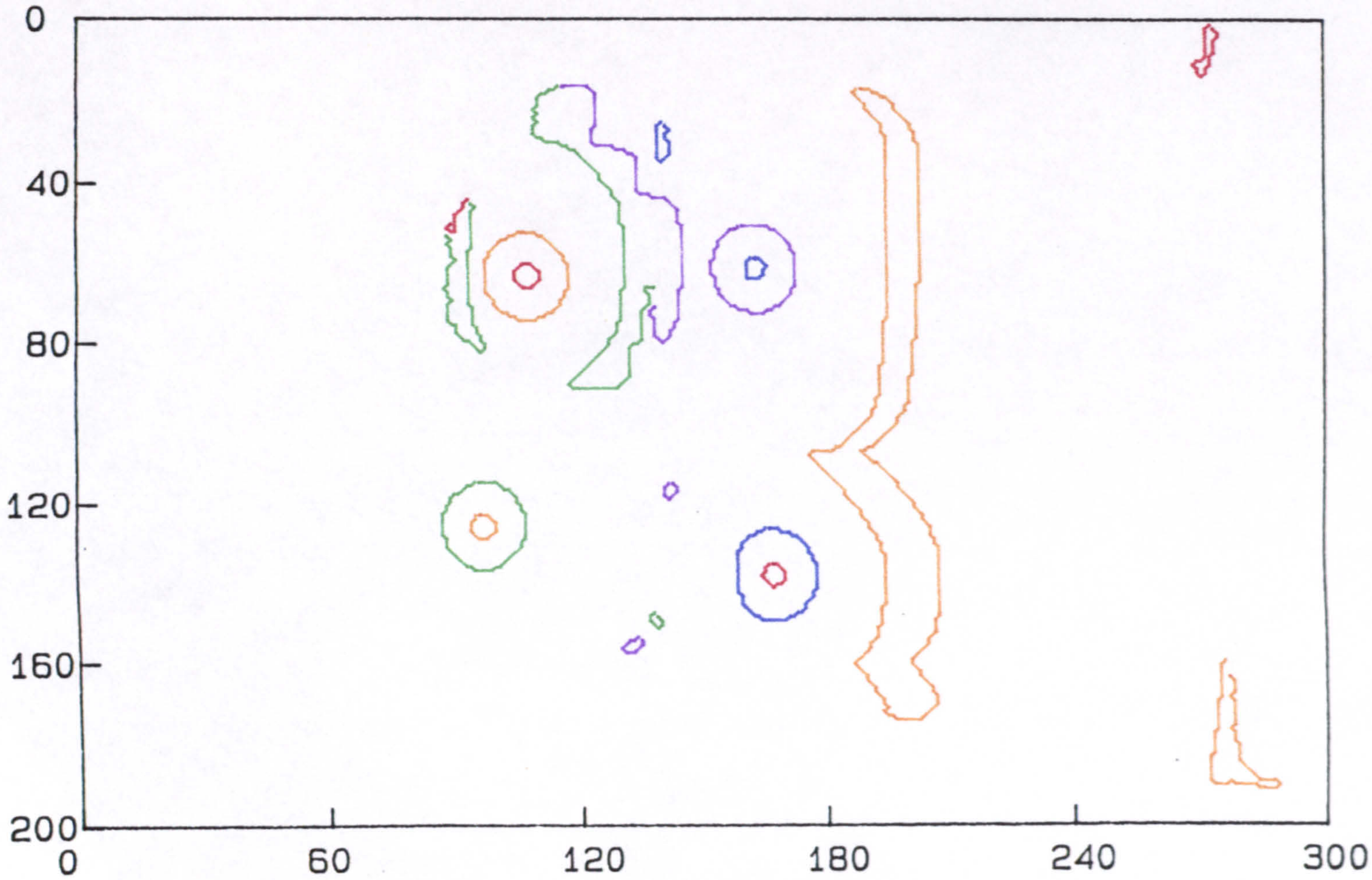
PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.60



PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.61



PLOT GENERATED BETWEEN LINE-SECTION CORNERS

Figure 12.62

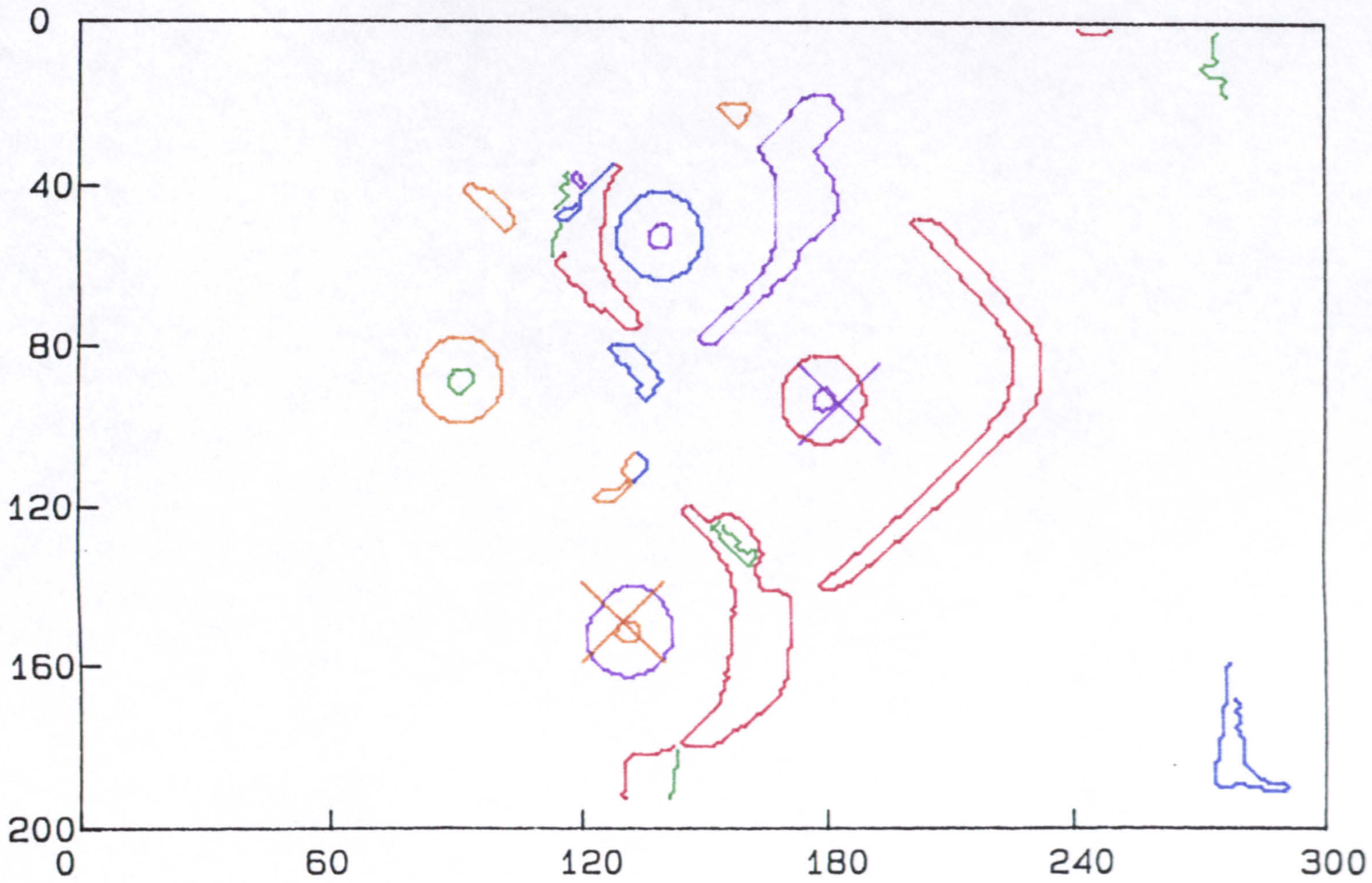


Figure AA1

```
LIST

10  REM  Software created for the British Airways Project Stage 1.
20  REM  This involves inspecting a robot loaded tray.
30  REM  The user is prompted to produce an ideal image.  A test image
40  REM  is then taken which is exclusive-ORed with the ideal.
50  REM  The resulting image is then transferred to the DUMC unit.
60  REM  Main program calling subroutines.
70  HOME
80 D$ =  CHR$ (4)
90  REM  Activate 80 Column card.
100  PRINT D$;"PR#3"
110  REM  Load parallel transfer subroutine.
120  PRINT D$;"BLOAD PARA"
130  REM  Load "AND" subroutine.
140  PRINT D$;"BLOAD AND"
150  REM  Load "EOR" subroutine.
160  PRINT D$;"BLOAD EOR"
170  PRINT "                    **  PROGRAM TO SELECT IDEAL IMAGE  **"
180  PRINT "": PRINT ""
190  HTAB 20
200  PRINT "Place empty tray on inspection "
210  HTAB 20
220  PRINT "station and watch the image."
230  PRINT "": PRINT ""
240  HTAB 20
250  PRINT "When the image is satisfactory"
260  HTAB 20
270  PRINT "press the black button on the"
280  HTAB 20
290  PRINT "digitizer paddle.
300  PRINT ""
310  HTAB 20
320  PRINT "If the image is satisfactory"
330  HTAB 20
340  PRINT "press Y key -else press N to"
350  HTAB 20
360  PRINT "continue digitization."
370  PRINT "": PRINT "": PRINT ""
380  HTAB 20
390  PRINT "(Press spacebar to continue.)"
400  GET A$
410  GOSUB 950: REM  Digitize an image into high-res. page 2.
420  TEXT
430  HOME
440  PRINT "": PRINT "": PRINT ""
450  PRINT "": PRINT ""
460  HTAB 20
470  PRINT "Now select another image of the"
480  HTAB 20
490  PRINT "same tray and press the black"
500  HTAB 20
510  PRINT "button to freeze frame again."
520  PRINT "": PRINT ""
530  HTAB 20
540  PRINT "If the image is satisfactory"
550  HTAB 20
560  PRINT "press Y,else press N to continue"
570  HTAB 20
580  PRINT "digitization."
590  PRINT "": PRINT "": PRINT "": PRINT ""
600  HTAB 20
610  PRINT "(Press spacebar to continue)"
```

Figure AA1 Continued.

```
620  GET A$
630  GOSUB 1050: REM  Capture an image and store in high-res page 1.
640  CALL 24630: REM  Call the image "ANDing" routine.
650  TEXT
660  HOME
670  VTAB 7
680  HTAB 20
690  PRINT "You are now ready to test."
700  HTAB 20
710  PRINT "Substitute the tray to be tested"
720  HTAB 20
730  PRINT "under the inspection station."
740  HTAB 20
750  PRINT "Press the button to take a "
760  HTAB 20
770  PRINT "test frame."
780  PRINT ""
790  HTAB 18
800  PRINT "(Press spacebar to continue.)"
810  GET A$
820  GOSUB 1050: REM  Digitize an image into high-res. page 1.
830  CALL 24576: REM  Call the EOR subroutine.
840  CALL 720: REM  Call the parallel transfer routine.
850  HOME
860  TEXT
870  HTAB 20
880  PRINT "Test sample taken and transmission completed."
890  HOME
900  PRINT "": PRINT "": PRINT ""
910  HTAB 18
920  PRINT "   For another sample press spacebar"
930  GET A$
940  GOTO 650
950  REM  Subroutine to digitize an image into Hi-Res page 2.
960  PRINT  CHR$ (17)
970  REM  POKE -16208,0:REM Set digitizer function byte.
980  HGR2
990  GOSUB 1180: REM  Subroutine to read the games paddles.
1000 Z =  PEEK ( - 16207): REM  Trigger digitizer.
1010  REM  M=PEEK(49249):REM Read the games button.
1020  IF M < 128 THEN  GOTO 990
1030  PRINT  CHR$ (18)
1040  RETURN
1050  REM  Subroutine to digitize an image into Hi-Res page 1.
1060  PRINT  CHR$ (17)
1070  POKE 49236,1: REM  Primary page.
1080  POKE 49232,1: REM  Graphics mode.
1090  POKE 49239,1: REM  Hi-Res graphics.
1100  POKE 49234,1: REM  All text or graphics.
1110  GOSUB 1180: REM  Subroutine to read the games paddles.
1120 Z =  PEEK ( - 16207): REM  Trigger digitizer.
1130 M =  PEEK (49249): REM  Read the games button.
1140  IF M < 128 THEN  GOTO 1070
1150  POKE 49234,1
1160  PRINT  CHR$ (18)
1170  RETURN
1180  REM  Subroutine to read the games paddles.
1190 Y =  PDL (0)
1200  IF Y > 255 THEN Y = 255
1210  POKE  - 16206,Y
1220  RETURN

]
```

Figure AA2

```
6004L

6004-   A9 40       LDA   #$40
6006-   8D 1A 60    STA   $601A
6009-   A0 00       LDY   #$00
600B-   A2 00       LDX   #$00
600D-   A9 20       LDA   #$20
600F-   8D 17 60    STA   $6017
6012-   8D 1D 60    STA   $601D
6015-   BD 00 40    LDA   $4000,X
6018-   5D 00 60    EOR   $6000,X
601B-   9D 00 40    STA   $4000,X
601E-   E8          INX
601F-   D0 F4       BNE   $6015
6021-   EE 17 60    INC   $6017
6024-   EE 1D 60    INC   $601D
6027-   EE 1A 60    INC   $601A
602A-   EA          NOP
602B-   AD 17 60    LDA   $6017
602E-   C9 40       CMP   #$40
6030-   D0 E3       BNE   $6015
6032-   EA          NOP
*L

6033-   EA          NOP
6034-   EA          NOP
6035-   60          RTS
6036-   A9 20       LDA   #$20
6038-   8D 4C 60    STA   $604C
603B-   A0 00       LDY   #$00
603D-   A2 00       LDX   #$00
603F-   A9 40       LDA   #$40
6041-   8D 49 60    STA   $6049
6044-   8D 4F 60    STA   $604F
6047-   BD 00 40    LDA   $4000,X
604A-   3D 00 20    AND   $2000,X
604D-   9D 00 40    STA   $4000,X
6050-   E8          INX
6051-   D0 F4       BNE   $6047
6053-   EE 49 60    INC   $6049
6056-   EE 4F 60    INC   $604F
6059-   EE 4C 60    INC   $604C
605C-   EA          NOP
605D-   AD 4C 60    LDA   $604C
*L

6060-   C9 40       CMP   #$40
6062-   D0 E3       BNE   $6047
6064-   EA          NOP
6065-   EA          NOP
6066-   EA          NOP
6067-   60          RTS
```

Figure AA3

```
 2D6L

02D6-    A9 00       LDA    #$00
02D8-    8D BA 03    STA    $03BA
02DB-    8D BB 03    STA    $03BB
02DE-    A9 FF       LDA    #$FF
02E0-    8D 93 C0    STA    $C093
02E3-    A9 02       LDA    #$02
02E5-    8D 9E C0    STA    $C09E
02E8-    A9 08       LDA    #$08
02EA-    8D 9C C0    STA    $C09C
02ED-    A9 58       LDA    #$58
02EF-    8D 91 C0    STA    $C091
02F2-    AE BA 03    LDX    $03BA
02F5-    BD C0 03    LDA    $03C0,X
02F8-    85 EC       STA    $EC
02FA-    BD A0 03    LDA    $03A0,X
02FD-    8D BC 03    STA    $03BC
0300-    EE BA 03    INC    $03BA
0303-    AE BA 03    LDX    $03BA
0306-    E0 19       CPX    #$19
0308-    F0 32       BEQ    $033C
*L

030A-    AE BB 03    LDX    $03BB
030D-    BD E0 03    LDA    $03E0,X
0310-    18          CLC
0311-    6D BC 03    ADC    $03BC
0314-    85 ED       STA    $ED
0316-    E8          INX
0317-    E0 09       CPX    #$09
0319-    30 08       BMI    $0323
031B-    A2 00       LDX    #$00
031D-    8E BB 03    STX    $03BB
0320-    4C F2 02    JMP    $02F2
0323-    8E BB 03    STX    $03BB
0326-    A0 00       LDY    #$00
0328-    AD 9D C0    LDA    $C09D
032B-    29 02       AND    #$02
032D-    F0 F9       BEQ    $0328
032F-    B1 EC       LDA    ($EC),Y
0331-    8D 91 C0    STA    $C091
0334-    C8          INY
0335-    C0 28       CPY    #$28
*L

0337-    30 EF       BMI    $0328
0339-    4C 0A 03    JMP    $030A
033C-    A9 01       LDA    #$01
033E-    8D 50 C0    STA    $C050
0341-    8D 52 C0    STA    $C052
0344-    8D 54 C0    STA    $C054
0347-    8D 57 C0    STA    $C057
034A-    60          RTS
```

Figure AA3 Continued.

```
 3A0L

03A0-   20 20 21    JSR   $2120
03A3-   21 22       AND   ($22,X)
03A5-   22          ???
03A6-   23          ???
03A7-   23          ???
03A8-   20 20 21    JSR   $2120
03AB-   21 22       AND   ($22,X)
03AD-   22          ???
03AE-   23          ???
03AF-   23          ???
03B0-   20 20 21    JSR   $2120
03B3-   21 22       AND   ($22,X)
03B5-   22          ???
03B6-   23          ???
03B7-   23          ???
03B8-   00          BRK
03B9-   FF          ???
03BA-   19 00 00    ORA   $0000,Y
03BD-   44          ???
03BE-   00          BRK
*L

03BF-   2C 00 80    BIT   $8000
03C2-   00          BRK
03C3-   80          ???
03C4-   00          BRK
03C5-   80          ???
03C6-   00          BRK
03C7-   80          ???
03C8-   28          PLP
03C9-   A8          TAY
03CA-   28          PLP
03CB-   A8          TAY
03CC-   28          PLP
03CD-   A8          TAY
03CE-   28          PLP
03CF-   A8          TAY
03D0-   50 D0       BVC   $03A2
03D2-   50 D0       BVC   $03A4
03D4-   50 D0       BVC   $03A6
03D6-   50 D0       BVC   $03A8
03D8-   AA          TAX
*L

03D9-   4C B5 B7    JMP   $B7B5
03DC-   AD 0F 9D    LDA   $9D0F
03DF-   AC 00 04    LDY   $0400
03E2-   08          PHP
03E3-   0C          ???
03E4-   10 14       BPL   $03FA
03E6-   18          CLC
03E7-   1C          ???
03E8-   AA          TAX
03E9-   60          RTS
03EA-   4C 51 A8    JMP   $A851
03ED-   EA          NOP
03EE-   EA          NOP
03EF-   4C 59 FA    JMP   $FA59
03F2-   BF          ???
03F3-   9D 38 4C    STA   $4C38,X
03F6-   58          CLI
03F7-   AD C0 D0    LDA   $D0C0
```

Figure AA4

```
]1

]LOAD DIG1
]LIST

10  REM  Program developed for British Airways project stage 2.
20  REM  A test image is captured and the raw binary image
30  REM  transferred directly to the DUMC Unit.
40  REM  Main program calling utility subroutines.
50  HOME
60 D$ =  CHR$ (4): REM  Control D
70  PRINT D$;"PR#3": REM  Activate 80-Column card.
80  PRINT D$;"BLOAD PARA": REM  Load parallel transfer routine.
90  PRINT "                    **  PROGRAM TO SELECT IDEAL IMAGE  **"
100  PRINT "": PRINT ""
110  HTAB 20
120  PRINT "Place empty tray on inspection "
130  HTAB 20
140  PRINT "station and watch the image."
150  PRINT "": PRINT ""
160  HTAB 20
170  PRINT "When the image is satisfactory"
180  HTAB 20
190  PRINT "press the black button on the"
200  HTAB 20
210  PRINT "digitizer paddle.
220  PRINT ""
230  HTAB 20
240  PRINT "If the image is satisfactory"
250  HTAB 20
260  PRINT "press Y key -else press N to"
270  HTAB 20
280  PRINT "continue digitization."
290  PRINT "": PRINT "": PRINT ""
300  HTAB 20
310  PRINT "(Press spacebar to continue.)"
320  GET A$
330  PRINT "under the inspection station."
340  HTAB 20
350  PRINT "test frame."
360  PRINT ""
370  HTAB 18
380  GOSUB 490: REM  Digitize into Hi-Res page 1.
390  CALL 720: REM  Parallel transfer subroutine.
400  HOME
410  HTAB 20
420  PRINT "Test sample taken and transmission completed."
430  HOME
440  PRINT "": PRINT "": PRINT ""
450  HTAB 18
460  PRINT "   For another sample press spacebar"
470  GET A$
480  GOTO 380
```

Figure AA4 Continued.

```
490  REM  Subroutine to digitize an image into Hi-Res page 1.
500  PRINT  CHR$ (17)
510  POKE 49236,1: REM  Primary page.
520  POKE 49232,1: REM  Graphics mode.
530  POKE 49239,1: REM  Hi-res graphics.
540  POKE 49234,1: REM  All text or grahics.
550  GOSUB 660: REM  Call subroutine to read games paddle.
560 Z =  PEEK ( - 16207): REM  Trigger digitizer.
570 M =  PEEK (49249): REM  Read games button.
580  IF M < 128 THEN  GOTO 510
590  POKE 49235,1: REM  Mix text and graphics.
600  PRINT "Is this satisfactory ?"
610  GET S$
620  IF S$ = "N" THEN  GOTO 510
630  POKE 49234,1
640  PRINT  CHR$ (18)
650  RETURN
660  REM  Subroutine to read the game paddles
670  REM  and set the digitizer thresholds.
680 Y =  PDL (0)
690  IF Y > 255 THEN Y = 255
700  POKE  - 16206,Y
710  RETURN

]
```

Figure AA5 Continued.

```
1000  REM  Hardware set-up phase.
1010  PRINT "     Entering the hardware set-up phase."
1015  PRINT
1020  PRINT "     Adjust the rotary table to the required starting orientation."
1025  PRINT
1030  PRINT "     To modify the binary capture thresholds use the game paddles."
1035  PRINT
1040  PRINT "     When the image is satisfactory press the button on the game paddle."
1045  PRINT
1050  PRINT "     If the image is satisfactory press the 'Y' key to enter the"
1055  PRINT
1060  PRINT "     sampling phase, else another key to continue digitization."
1065  PRINT : PRINT : PRINT
1070  PRINT "                    Press Spacebar to Continue."
1075  PRINT
1080  GET A$
1100  REM  Set-up digitization proceedure.
1110  PRINT  CHR$ (17)
1120  POKE 49236,1: REM  Primary page.
1130  POKE 49232,1: REM  Graphic mode.
1140  POKE 49239,1: REM  Hi-Res mode.
1150  POKE 49234,1: REM  All graphics.
1160  REM  Read paddle.
1170  GOSUB 1300
1180 Z =  PEEK ( - 16207): REM  Trigger the digitizer.
1190 M =  PEEK (49249): REM  Read paddle button.
1200  IF M < 128 THEN  GOTO 1160
1210  POKE 49235,1
1215  PRINT  CHR$ (18)
1220  PRINT "          Is this satisfactory?"
1230  GET A$
1240  IF A$ = "N" THEN  GOTO 1160
1260  PRINT  CHR$ (18)
1270  RETURN
1300  REM  Subroutine to read paddles.
1310 X =  PDL (0):Y =  PDL (1) + X
1320  IF Y > 255 THEN Y = 255
1330  POKE  - 16206,X: REM  Write to digitizer threshold registers.
1340  POKE  - 16205,Y
1350  RETURN
1400  REM  Sampling digitization.
1410  PRINT  CHR$ (17)
1420  POKE 49236,1
1430  POKE 49232,1
1435  POKE 49239,1
1440  POKE 49234,1
1445 Z =  PEEK ( - 16207): REM  Trigger the digitizer.
1447  PRINT  CHR$ (18)
1450  RETURN
1500  REM  Routine to rotate the table through the specified angle.
1510  POKE 49298,128: REM  Set up the DDRB
1520  POKE 49307,128: REM  Set up the ACR
1530  POKE 49310,0: REM  Disable the interrupt.
1540  FOR S = 1 TO NS
1550  REM  Set up the high-order counter byte to trigger timer.
1560  POKE 49301,78
1565  FOR WW = 0 TO 200: NEXT WW
1570  NEXT S
1580  RETURN
1700  REM  Hold loop to allow 68020 disc chang.
1710  PRINT "          30 Images transferred to 68020 unit: change disc if necessary."
1730  FOR T = 0 TO 4
1740  FOR WW = 0 TO 1000
1750  NEXT WW
1760  PRINT  CHR$ (7);
1770  NEXT T
1780  GET T$
1790  RETURN
```

Figure AA5

```
]LOAD LEARN
]LIST

100  REM  Progam to coordinate the object learning operations.
110 D$ =  CHR$ (4): REM  Control D
120  PRINT D$;"PR£3": REM  Enable digitizer/80 column card.
130  PRINT D$;"BLOAD PARA": REM  Load parallel interface routine.
140  HOME
145  PRINT : PRINT : PRINT : PRINT
150  PRINT "                  ********************************************"
160  PRINT "                  *                                          *"
170  PRINT "                  *     British Airways Image Capture Software   *"
175  PRINT "                  *                                          *"
180  PRINT "                  * for the learning phase in object recognition.*"
190  PRINT "                  *                                          *"
200  PRINT "                  ********************************************"
210  PRINT : PRINT : PRINT
220  PRINT "                  The progam starts with a set-up phase in which"
230  PRINT "                  the digitizer thresholds are set and the rotary"
240  PRINT "                  table orientated."
245  PRINT
250  PRINT "                              The processing continues with the image"
260  PRINT "                  phase in which the specified number of images are"
270  PRINT "                  captured at the required angular interval=."
300  REM  Set-up phase option.
305  PRINT
310  PRINT "       Enter 1 if the set-up phase is required."
315  PRINT
320  PRINT "       Enter 2 if the set-up phase is not required."
325  PRINT : PRINT
330  GET OP$
335  PRINT
340  IF OP$ = "1" THEN  GOSUB 1000: REM  Set up digitization hardware.
350  PRINT "       About to enter sampling phase."
355  PRINT
360  PRINT "       Enter the number of images to be captured."
365  PRINT
370  INPUT NI
380  PRINT "       Enter the number of 0.9 degree interval= to be made per step."
385  PRINT
390  INPUT NS
400 AS = NS * 0.9
410  PRINT "       Angle of rotation per step = "AS
420  PRINT : PRINT
500  REM  Main sampling loop.
510  FOR N = 1 TO NI
520  IF N = 1 THEN  GOTO 540
530  REM  Rotate the rotary table.
535  GOSUB 1500
537  REM  Read paddle to set threshold.
538  GOSUB 1300
540  REM  Digitize into page 1.
550  GOSUB 1400
560  POKE 49235,1: REM  Mix text and graphics.
570  PRINT "          Unit in wait state."
580  PRINT "          Image about to be transferred to MC6800 Unit is number "N"."
582  IF N = 1 THEN  GOTO 600
590  IF N = 61 THEN  GOSUB 1700
591  IF N = 121 THEN  GOSUB 1700
592  IF N = 91 THEN  GOSUB 1700
593  IF N = 151 THEN  GOSUB 1700
594  IF N = 181 THEN  GOSUB 1700
595  IF N = 31 THEN  GOSUB 1700
597  FOR WW = 0 TO 1000: NEXT
600  REM  Call parallel transfer routine.
610  CALL 720
615  POKE 49235,1: REM  Mix text and graphics.
620  PRINT "          Image transfer complete.'
625  PRINT  CHR$ (7)
630  NEXT N
640  PRINT : PRINT : PRINT
650  PRINT "          All "NI" images have been captured and tranferred."
670  PRINT "          Angle of rotation between each subsequent image is =- "AS" degrees.
680  FOR T = 0 TO 3
682  FOR WW = 0 TO 10000
683  FOR W = 0 TO 100: NEXT W
684  NEXT WW
685  PRINT  CHR$ (7)
690  NEXT T
700  END
```

Aug 4 15:34:15 1986 acofbo.c Page 1

Figure AD1

```
#include <std.h>          /* inclusion of standard function libraries */
char screen[7681]{0};     /* preliminary storage array for image */
char pixel[53761]{0};     /* expanded storage array for image */
int printo=0;             /* print option variable */
int ufc=0;                /* coordinate in-output and VDU presentation variable */
int nel[9]{0};            /* area element number array */
int ncount[10]{0};         /* area count array */
int cn=5;                 /* number of areas counted -default 5 */
int thr[10]={0,328,503,891,255,912,0,0,0,0}; /* default area threhold values  */
int val[5][9]={     /* array of default corner coordinates for search windows */
     {0,1,2,3,4,5,6,7},
     {0,55,125,233,81,253,0},
     {0,0,0,0,87,62,0},
     {0,71,156,268,110,279,0},
     {0,78,78,48,182,168,0},
               };
char nstring[]="five";
main()/* whitesmiths compiler version for selective area count- */
{ /* main function coordinating calling sequence of processing functions */
GLOBAL int printo,ufc;
int c,r,b,d,e;
modithr(); /* fct. to allow user to set thresholds,printer dump,screen o/p opt. */
for(k=0;k<1;k++){
   great();        /* greetings screen display */
   aquidw();       /* fct. to read apple image data into primary storage array */
   bitmatdw();     /* expansion function to convert the above array into a one image bit per byte f
   if(printo==1){ /* call to dot matrix image printing routine if print option variable is set */
      pridw();  }
   switch(ufc){   /* conditional branch on set ufc option */
         case 1 : ufcf();break;/* counting routine with short screen listing */
         case 2 : ufcoudw();break;/* counting routine with counts and additional information liste
         case 3 : ufcw();break;/* fct. allowing keyboard specifiction of counting windows */
               }
   decow();/* decision making fct. based on counts calculated in functs above.*/
   putfmt("\n                Press \"a\" key to run program again. \n");
   if (getch()=='a'){k= -1};getch();getch();/* Dummy reads */
               }
}
```

Aug 4 15:55:05 1986 agrf.c Page 1

Figure AD3

```
#include <std.h>
greet() /* intro for British Airways demo */
{
int key;
putfmt("  \n");
putfmt("\n\n\n\n");
putfmt("          ************************************************\n");
putfmt("          *                                              *\n");
putfmt("          *          68000  INSPECTION  SOFTWARE         *\n");
putfmt("          *                                              *\n");
putfmt("          *  For DURHAM UNIVERSITY ENGINEERING DEPARTMENT *\n");
putfmt("          *                                              *\n");
putfmt("          *          BRITISH  AIRWAYS  PROJECT           *\n");
putfmt("          *                                              *\n");
putfmt("          *                    Created  September 1984   *\n");
putfmt("          *                                              *\n");
putfmt("          ************************************************\n");
/*        getfmt("%i",&key);  */  /* any key press to continue */
}
```

Figure AD2

```
#include <std.h>
modithr()/* function to modify thresholds - set print option.*/
         /* set search area input output options              */
{
GLOBAL int printo,ufc,cn;
GLOBAL int val[5][8],thr[10];
int m,i,win,xt,yt,xb,yb,an;
short f,t;
f=1;
   putfmt("Do you wish to have print option ?\n");
   if(getch()=='y'){
     printo=1;
                   }
else printo=0;
getch();getch();/* dummy reads */
putfmt("Which ufc option do you require ?\n\n");
putfmt("Enter 1 for ufcf (few prints and fast).\n");
putfmt("Enter 2 for ufcouw (prints coords etc).\n");
putfmt("Enter 3 for ufcw (reads coords from stdin).\n");
getfmt("%i",&ufc);
putfmt("How many areas do you wish to count ? (default = 5.) \n");
getfmt("%i",&cn);
putfmt("Do you want to modify thresholds ?\n");
if(getch()=='y'){ getch();getch();/* Dummy reads */
  putfmt("      Threshold modification stage.\n\n");
  for(n=1;n<=cn;n++){
  putfmt("     Default value of thr%i= %i - is this satisfactory ? \n\n",n,thr[n]);
     if(getch()=='n'){
      getch();getch();
      do{
      putfmt("    Enter new value of thr%i    :  \n",n);
      getfmt("%i",&thr[n]);
      putfmt("    New value of thr%i =  %i \n",n,thr[n]);
      putfmt("    Is this satisfactory ?\n");
      if (getch()=='n'){f=0};
      else {f=1};
      getch();getch();
       }
      while(f !=1);
                 }
else { getch();getch(); }
                 }
            }
else { getch();getch(); }
putfmt("Do you want to modify count window ?\n");
if(getch()=='y'){
  getch();getch();/* dummy reads */
  putfmt("How many windows do you wish to modify ?\n");
  getfmt("%i",&win);
  for(n=1;n<=win;n++){
     putfmt("Enter particular area number.\n");
     getfmt("%i",&an);
     putfmt("Enter top left corner x,y coordinates.\n");
     getfmt("%i%i",&xt,&yt);
     putfmt("Your two upper left coordinates are now :%i \t %i\n",xt,yt);
     putfmt("Enter bottom right corner x,y coordinates.\n");
```



```
             getfmt("%i%i",&xb,&yb);
             putfmt("Your two bottom right coordinates are now :%i \t %i\n",xb,yb);
             val[1][an]=xt;
             val[2][an]=yt;
             val[3][an]=xb;
             val[4][an]=yb;
                          }
                    }
       else {
         getch();getch();
            }
}
```

Aug 4 16:55:05 1986 aaltaquip.c Page 1

Figure AD4

```
#include <std.h>
aquipw() /* to get image from Apple by parallel interface link and place in a 1D pointer array
{
extern char screen[7681];
int ipar(),getpar(),tstpar();
short x;
register unsigned short y;
register char *pscreen;
pscreen=screen;
ipar();/* initialize on-board 68300 VIA */
        for(k=0;k<1;){
           if((getpar())=='X'){ /* test single character for start char */
             k++;
             for(y=0;y<7680;y++){
                *(pscreen++)=getpar(); /* read in digital image into linear array */
                                  }
                               }
                     }
}
```

Aug 4 16:57:14 1986 aaabitaa.c Page 1

Figure AD5

```
#include <std.h>
bitmatch() /* to convert 8K array into one byte per pixel 54< array */
{
GLOBAL char screen[7681];
GLOBAL char mpixel[53761];
register char *pmpixel;
register char *pscreen;
register unsigned short mask;
register unsigned short j;
pmpixel=mpixel;
pscreen=screen;
            for(j=1;j<7681;j++,pscreen++){
                 for(mask=1;mask !=0x80;mask <<=1){
                     if ((*pscreen & mask)!=0)
                         *pmpixel++=1;
                     else{
                         *pmpixel++=0;
                         }
                                                  }
                                }
                    putfmt("transfer into bits complete \n");
}
```

Aug 4 16:56:05 1986 aufcf.c Page 1

Figure AD6

```
#include <std.h>
ufcf() /* counting routine displaying only the counts for respective areas */
{
GLOBAL char mpixel[53760];
GLOBAL int ncount[10];
GLOBAL int nel[9];
GLOBAL int val[5][8];
GLOBAL char nstring[];
GLOBAL int cn;
short t;
int xt,xxt,yt,yyt,a1,xb,yb,count;
int *pncount;
register char *tpmpixel;
char *pmpixel;
pncount=ncount+1;
        for(t=1;t<=cn;t++){
          pmpixel=mpixel;count=0;yyt=yt;
          xt=val[1][t];
          yt=val[2][t];
          xb=val[3][t];
          yb=val[4][t];
                for(pmpixel=pmpixel+(280*yt);yyt<=yb;yyt++,pmpixel+=280){
                   tpmpixel=pmpixel;xxt=xt;
                   for(tpmpixel=tpmpixel+xt;xxt<=xb;xxt++){
                      if ((*tpmpixel++)>0){ count++; }
                                                        }
                                                              }

                    *pncount++=count;
                          }
          pncount=ncount+1;
          for(t=1;t<=cn;t++){
           putfmt("          Areas corresponding are :-   %i\n",*pncount++);
                        }
          putfmt("\n\n");
}
```

Aug 4 16:55:05 1986 adef.c Page 1

Figure AD7

```
#include <std.h>
decpw() /* decision making based on counts calculated in previous function */
{
GLOBAL int ncount[10],nel[9],tar[10];
GLOBAL int cn;
int al;
        for(al=1;al<=cn;el++){
           if(ncount[el]<tar[el]){
             putfmt("     Item %i absent.\n\n",el);
                                 }
           else{
               putfmt("Item %i is present and located.\n\n",el);
               }
                             }
}
```

Figure AD8


```
#include <std.h>                    /* incluce stancard Whitesmith's header */
/* declaration arc initialization of external variables */
short *ppcint=point;                /* line element pcinter  */
char *PMPIXEL=mpixel;               /* image element pcinter */
short *line[100]=pcirt;             /* array of pointers to line starts */
short *endline[100]=point;          /*                      line encs   */
short *aline[100]=pcirt;            /* alterrative          line starts */
short *aendline[100]=pcint;         /* alternative          line encs   */
char mpixel[53761]=0;               /* exparded image array             */
short cimpoint[100][25]=0;          /* line number incexed feature array*/
short impoint[100][10]=0;           /*              imporant line points */
short timpoint[500][15]=0;          /*                    output features*/
short spoint[300]=0;
char screen[7681]=0;                /* small received image array       */
short point[30000]=0;               /* secuential line storage array    */
short bound[100]=0;                 /* line indexed bcundary point array*/
short totarea[100]=0;               /*              area          array */
short totperi[100]=0;               /*              perimeter            */
short paction[100]=0;               /*              compaction           */
unsigned short type[100][3]=0;      /*              shape type           */
short outline[100][30]=0;           /*                                   */
short minpid=0;                     /*   important points                */
short midpid=0;
short maxpid=0;
short LINE=0;                       /* line index                        */
short FI=0;                         /* frame incex                       */
short J=0;                          /*   sundry variables                */
short I=0;
int pindex=0;
int LS=0;
short N=0;
short BP=0;
short fd=0;
short impl=0;
short lilen=50;                     /* optical minimum line length considerec */
struct FILEDATA                     /* DOS file structure support        */
        {
        char name[15];
        int block;
        int load;
        int start;
        int count;
        char *pbuf;
        int fsize;
        };
struct FILEDATA fdata= {
        "**************",
        0,0,0,15,NULL,0
                        };
struct FILEDATA *fss=NULL;
char fname[16]="**************";
int ni=0;                           /* number of images */
int namin=0;                        /* index to file name */
int mult=0;                         /* interval in file incex */
int RI=1;                           /* main loop index */
short crossind=0;                   /* centre cross variables */
```



```
short cross[10][4][2][14]=0;

/* External variables used in setting program options */
short IMSOURCE=0;                   /* test image source option          */
short PRINTOR=0;                    /* print original image option       */
short PRINTHIN=0;                   /* print thirred image option        */
short PRINTFT=0;                    /* print feature table option        */
short PLOTCORN=0;                   /* corner image plotting option      */
short EOS=0;                        /* end of secuence marker            */
short FFCEST=0;                     /* feature file cestiration flag     */
```

```
main()          /* coordinating function for full scene analysis from disc      */
                /* image storage on 68020 system with feature transfer to cuet. */
{
CLOBAL short IMSOURCE,PRINTCR,PRINTHIN,PRINTFT,FFCEST;
GLOBAL short minpic,midpid,maxpid,J,I,N,BP,fd,lilen;
GLOBAL int pindex,RI,ni;
GLOBAL short bound[100];
GLOBAL short totperi[100];
GLOBAL short totarea[100];
GLOBAL short paction[100];
GLOBAL short cimpoint[100][25];
GLOBAL short impcint[100][10];
GLOBAL unsigned short type[100][3];
GLOBAL short spoint[300];
GLOBAL short outline[100][30];
GLOBAL short point[30000];
int aquipwd();
int c,m;
short s,n;
char getch();
int selshad();
cache();                        /* set cache running on 68020 board */
initdu();                       /* initialize 68020 on-board UART */
greet();
changep();
if(IMSOURCE==1){
  filepar();                    /* read in the file processing details */
               }
if(IMSOURCE==2){
     apar();                    /* reac in the apple processing details */
               }
for(RI=1;RI<=ni;RI++){
   if(IMSOURCE==1){
     /* if((RI ==31 )||(RI == 52)||(RI == 81)){ for bowl file access */
       if((RI ==31)||(RI==61)||(RI==91)||(RI==121)||(RI==151)||(RI==181)){
       putfmt("Charge the disc if necessary.\n");
       getch();getch();
                                                                        }
                  }
for(c=0;c<100;c++){             /* reinitialization of global variables */
   bound[c]=0;
   totperi[c]=0;
   totarea[c]=0;
   paction[c]=0;
   for(m=0;m<3;m++){
      type[c][m]=0;
```



```
                   }
   for(m=0;m<25;m++){
      cimpoint[c][m]=0;
                    }
   for(m=0;m<10;m++){
      impoint[c][m]=0;
                    }
   for(m=0;m<30;m++){
      type[c][m]=0;
      outline[c][m]=0;
                    }
                  }
for(ppoint=point,m=0;m<30000;m++){
   *(ppoint++)=0;
                                 }
for(m=0;m<300;m++){
   spcint[m]=0;
                  }
aline[0]=point;
endline[0]=pcint;
aercline[0]=pcint;
pocint=point;
PMPIXEL=moixel;
line[0]=point;
LINE=0;
minpid=0;
micpid=0;
maxpid=0;
J=0;
I=0;
pirdex=0;
N=0;
BP=0;
fd=0;
```

Figure AD8 Continued.

Figure AD8 Continued.

```
/* putfmt("Bourdary threshold modification completed.\n\n"); */
if(IMSOURCE==1){
  if(aquirwd()==(-1)){ /* Test image read error */
    goto terminate;
                     }
  putfmt("Disk image received.\n\n");
                   }
else if(IMSOURCE==2){
       aquirwap();
       putfmt("Image received from Apple.\r\n");
                    }
     bitmatpw();
     putfmt("Expansion of bits into bytes completed.\n\n");
     if(PRINTOR==1){
       pripw();
                   }
     kon();
     putfmt("Edge extraction completed.\n\n");
     if(PRINTHIN==1){
       pripw();
                    }
     for(N=0;(((s=selshad()) !=0)&&(N !=100));N++){
```



```
       putfmt(".");
                                                  }
     putfmt("\n");
     putfmt("Chain coding sequence completed.\r\n");
     fclosed();
     putfmt("Test for closed outlines completec.\n\n");
     xend();
     putfmt("Artificial end of boundary generatior completed.\n\n");
     fperi();
     putfmt("Closec tcundary perimeter calculation completed.\n");
     farea();
     putfmt("Closed tcundary area calculation completed.\n");
     compact();
     putfmt("Closec tcuncary compaction-factor calculation completed.\n\n");
     shadw();
     putfmt("Oper line feature point derivation completed.\n\n");
     cimp();
     putfmt("Cper line feature calculaticn completed.\n");
     ccircle();
     putfmt("Centre circle coordinates calculated.\n");
     dcentre();
     putfmt("Cistarces to centre circle calculated.\n");
     timp();
     putfmt("Output feature arranging routine finished.\n");
     if(PRINTFI==1){
       printable();
                   }
     if(PLOTCORN==1){
            shipccrnpw();
            putfmt("Press <CR> to ccntinue.\n");
            getch();getch();
                        }
                  }
     if(FFCEST==1){
       toduet();
                   }
     toffile();  /* Create a feature data file */
   putfmt("Transfer of all image features completed.\n");
   exit();
   terminate:
   putfmt("\rPrograw terminated due to cisk read error.\n");
   exit();
   }
```

Figure AD9


```
#include <std.h>
changep() /* Program operating parameters and options set-up section. */
{
GLOBAL short lilen;
GLOBAL short ECS,IMSOURCE,PRINTOR,PRINTHIN,PRINTFT,PLOTCORN,FFDEST;
short actlen;
char getch();
putfmt("       ************************************************************\n");
putfmt("       *                                                          *\n");
putfmt("       *                  PROGRAM SET-UP PHASE                    *\n");
putfmt("       *                                                          *\n");
putfmt("       ************************************************************\n");
putfmt("\n\n\n\n");
putfmt("Program Parameter Set-up.\n\n");
putfmt("Enter boundary outline-length threshold parameter:\n");
getfmt("%s",&actlen);
putfmt("Any line below %s boundary points is discarded in subsequent processing stages.\r\n",actlen);
lilen=4*actlen;
putfmt("Program Display Option Set-up.\n\n");
while((IMSOURCE !=1)&&(IMSOURCE !=2)){
putfmt("Are the images to be read in from disk or from the Apple?\n\n");
putfmt("Enter 1 if the images are to be read in from disk drive 0.\n");
putfmt("Enter 2 if the images are to be read in from the Apple.\n");
getfmt("%s",&IMSOURCE);
                                    }
putfmt("\n");
if(IMSOURCE==1)putfmt("The images are to be read in from disk file.\n");
else putfmt("The images are to be received from the Apple.\n");
putfmt("Do you want to output feature data to CLET?\n");
if(getch()=='y'){
  FFDEST=1;
  putfmt("Feature data will also be output to the CLET.\n\n");
                }
getch();getch();
putfmt("\n");
putfmt("Is a dot-matrix printout of initial image required?\n");
if(getch()=='y'){
  putfmt("A dot-matrix printout of original image will be produced.\n\n");
  PRINTOR=1;
                }
getch();getch();
putfmt("Is a dot-matrix printout of the thinned/edge-extracted outline required.\n");
if(getch()=='y'){
  putfmt("A dot-matrix printout of edge-extracted image will be produced.\n");
  PRINTHIN=1;
                }
getch();getch();
putfmt("\n");
putfmt("Is a printout of the feature-table data required?\n");
if(getch()=='y'){
  putfmt("A feature-table printout will be provided.\n");
  PRINTFT=1;
                }
getch();getch();
putfmt("\n");
putfmt("Is a presentation of the chain-coded outlines required on the DUET?\n");
```



```
if(getch()=='y'){
  putfmt("Chain-coded object outlines will be represented on the DUET.\n");
  PLOTCORN=1;
                }
getch();getch();
putfmt("\n");
putfmt("Is end-of-plot sequence marker to be sent for each complete plot transferred.\n");
  if(getch()=='y'){
    EOS=1;
                  }
getch();getch();
putfmt("\n");
}
```

Figure AD10

```
Feb 15 15:56:18 1988  fileparf.c Page 1

#include<std.h>
filepar()
{
GLOBAL int ni,namin,mult;
GLOBAL char fname[3][16];
char bname[7];
int n;
for(n=0;n<15;n++){
   frame[0][n]='+';
                  }
frame[0][15]='\0';
putfmt("        ***************************************************\n");
putfmt("        *                                                 *\n");
putfmt("        *          DISK FILE INPUT PARAMETERS STAGE       *\n");
putfmt("        *                                                 *\n");
putfmt("        ***************************************************\n");
putfmt("\n\n");
putfmt("Enter the number of file images to be accessed on disc.\n");
getfmt("%i",&ni);
putfmt("Enter the base name of the first file in the secuence.\n");
getfmt("%p",&bname[0]);
for(n=0;n<6;n++){
   fname[0][n]=bname[n];
                  }
putfmt("Enter the start index of file with this base name.\n");
getfmt("%i",&namin);
putfmt("Enter the file interval index.\r");
getfmt("%i",&mult);
putfmt("Accessing %i files from file %p, with base index %i, \nat %i unit intervals.\n"
,ni,&fname[0][0],namin,mult);
}
```

Figure AD11

```
Feb 15 10:31:24 1988  apar.c Page 1

#include<std.h>
apar()
{
GLOBAL int ni;
putfmt("        ***************************************************\n");
putfmt("        *                                                 *\n");
putfmt("        *          APPLE IMAGE INPUT PARAMETERS STAGE     *\n");
putfmt("        *                                                 *\n");
putfmt("        ***************************************************\n");
putfmt("\n\n");
putfmt("Enter the number of file images to be read in from the APPLE.\n");
getfmt("%i",&ni);
putfmt("Accessing %i images from the APPLE.\n\n",ni);
}
```

```
#include<std.h>
struct FILEDATA
        {
        char name[16];
        int block;
        int load;
        int start;
        int count;
        char *pbuf;
        int fsize;
        };
aquipwd() /* Reads in specified image files from disc */
{
extern char screen[7681];
extern int namin,mult,RI;
extern struct FILEDATA fdata2,*fss2;
extern char fname[3][16];
int n,drive,k;
char *pframe,*pscreen;
fss2 = &fdata2;
drive=0;
if((n=system(7,&drive))!=0){
  putfmt("Error in accessing drive 0.\n");
  goto end;
                           }
if((n=system(16))!=0){
  putfmt("Error in restoring drive 0.\n");
  goto end;
                     }
/* Generate a file name */
pframe=(&fname[0][6]);
if(RI != 1){
  namin += mult;
          }
cecccf(pframe,3,"%- i",namin);
putfmt("File name generated is %p.",&fname[C][C]);
for(n=0;n<15;n++){ /* Leave the null termination. */
   fdata2.name[n]=fname[0][n];
                  }
if((n=system(10,fss2))!=0){
  putfmt("Error in opening file for read.\n");
  goto end;
                          }
fdata2.pbuf=screen;
fdata2.count=15;
if((n=system(13,fss2))!=0){
  putfmt("Error in reacing file.\n");
  goto end;
                          }
system(8);
putfmt("Drive 0 deselected.\n");
return(0); /* Return success indication */
end:
system(8);
putfmt("Drive 0 deselected.\n");
return(-1); /* Return failure */

}
```

Figurc AD12

Figure AD13a


```
#include <std.h>
kon()/* thinning and edge extraction algorithm- edge outside shadow outline version */
{
GLOBAL char mpixel[53760];
register short y,i;
int t;
register char *pmpixel;
pmpixel=mpixel;
             /* Testing of four corners */
             if(*pmpixel !=0){   /*top left*/
              if((*(pmpixel+1)==1) && (*(pmpixel+280)==1)){
               *pmpixel=2;
                                                          }
                                 }
             if(*(pmpixel+279)!=0){  /* top right */
              if((*(pmpixel+278)==1)&& (*(pmpixel+559)==1)){
               *(pmpixel+279)=2;
                                                           }
                                   }
             if(*(pmpixel+53480)!=0){  /* bottom left */
              if((*(pmpixel+53481)==1)&& (*(pmpixel+53200)==1)){
               *(pmpixel+53480)=2;
                                                              }
                                     }
             if(*(pmpixel+53759)!=0){  /* bottom right */
              if((*(pmpixel+53758)==1)&& (*(pmpixel+53479)==1)){
               *(pmpixel+53759)=2;
                                                              }
                                     }
```

Figure AD13a Continued.

```
   pmpixel++;
            for(i=1;i<279;i++){/* Testing along top row */
             if(*pmpixel ==1){
              if((*(pmpixel-1)>0) && (*(pmpixel+1)>0) && (*(pmpixel+280)>0)){
               *(pmpixel)=2;

                                                                    }
                                                                  }
             pmpixel++;
                                 }
 pmpixel=mpixel+53481;  /* Testing along bottom row */
           for(i=1;i<279;i++){
            if(*pmpixel ==1){
             if((*(pmpixel-1)>0) && (*(pmpixel+1)>0) && (*(pmpixel-280)>0)){
              *(pmpixel)=2;

                                                                  }
                                }
            pmpixel++;
                                 }
cmpixel=mpixel+280;  /* Testing down first column */
          for(y=1;y<191;y++){
           if(*cmpixel !=0){
            if((*(cmpixel+1)>0) && (*(pmpixel-280)>0) && (*(pmpixel+280)>0)){
             *cmpixel=2;

                                                                  }
                          }
              pmpixel=pmpixel+280;
                          }
```



```
cmpixel=mpixel+559;  /* Testing along final column */
          for(y=1;y<191;y++){
           if(*cmpixel !=0){
            if((*(pmpixel-1)>0) && (*(pmpixel-280)>0) && (*(pmpixel+280)>0)){
             *pmpixel=2;

                                                                  }
                          }
              pmpixel=pmpixel+280;
                          }
     /* Main body of program-testing central image block */
          pmpixel=mpixel+281;
          for(y=1;y<191;y++){
             for(i=1;i<279;i++,pmpixel++){
                if(*(pmpixel)==1){
                  /* testing of four-connected neighbours */
                  if((*(pmpixel+1)==0) || (*(pmpixel-1)==0) || (*(pmpixel-280)==0) || (*(pmpixel+280)==0))
                    /* labelling process to remove points from image */
                    else *pmpixel=2;
                                }
                                    }
          pmpixel=pmpixel+2; /* to bring down to next row */
                             }
/* Loop to change 2s to 0s -conversion to remove labelled point from array */
             for(pmpixel=mpixel,t=0;t<53761;t++,pmpixel++){
                  if(*pmpixel==2){
                     *pmpixel=0;
                                 }
                  if(*(++pmpixel)==2){
                     *pmpixel=0;
                                     
```

Figure AD13b

```
        .text
        .even
.kon:
        move.l  #107,d0
        trap    #15
        .word   2
        movem.l d5/d4/a3,-(sp)  * registers to be preserved
        move.l  #.mpixel,a3     * testing top left corner
        tst.b   (a3)
        bne.s   topr
        cmpi.b  #1,1(a3)
        beq.s   set1
        cmpi.b  #1,290(a3)
        beq.s   set1
        bra.s   topr
set1:
        move.b  #2,(a3)
topr:
        move.l  #98,d0
        trap    #15
        .word   2
        tst.b   279(a3)
        bne.s   botl
        cmpi.b  #1,278(a3)
        beq.s   set2
        cmpi.b  #1,559(a3)
        beq.s   set2
        bra.s   botl
set2:
        move.b  #2,279(a3)
botl:
        move.l  #99,d0
        trap    #15
        .word   2
        move.l  a3,a2
        adda.l   #53480,a2
        tst.b   (a2)
        bne.s   botr
        cmpi.b  #1,1(a2)
        beq.s   set3
        cmpi.b  #1,-280(a2)
        beq.s   set3
        bra.s   botr
set3:
        move.b  #2,(a2)
botr:
        move.l  #100,d0
        trap    #15
        .word   2
        adda.l   #279,a2
        tst.b   (a2)
        bne.s   trow
        cmpi.b  #1,-1(a2)
        beq.s   set4
        cmpi.b  #1,-280(a2)
        beq.s   set4
```



```
        bra.s   trow
set4:
        move.b  #2,(a2)
trow:
        move.l  #101,d0
        trap    #15
        .word   2
        move.l  a3,a2
        moveq   #1,d4
npl:
        addq.w  #1,d4
        cmpi.w  #279,d4
        bge.s   brow
        addq.l  #1,a2
        tst.b   (a2)
        bne.s   npl
        moveq   #0,d3
        cmpi.b  #1,-1(a2)
        beq.s   tr1
        ori.b   #2,d3
```

Figure AD13b Continued.

```
set8:
        move.b  #2,(a2)
        bra.s   np4
main6:
        move.l  #42,d0
        trap    #15
        .word   2
        move.l  a3,a2
        adda.w  #278,a2
        moveq   #1,d5           * Row index
+nrow:
        addq.w  #1,d5
        cmpi.w  #192,d5
        bge.s   chlto0
        moveq   #1,d4           * Column index
        addq.l  #2,a2
+ncol:
        addq.l  #1,d4
        cmpi.l  #280,d4
        bge.s   +nrow
        addq.l  #1,a2
        tst.b   (a2)
        bne.s   +ncol
        moveq   #0,d3           * NL
        cmpi.b  #1,-280(a2)
        beq.s   J1
        ori.b   #1,d3
J1:     cmpi.b  #1,-1(a2)
        beq.s   J2
        ori.b   #2,d3
J2:     cmpi.b  #1,1(a2)
        beq.s   J3
        ori.b   #4,d3
J3:     cmpi.b  #1,280(a2)
        beq.s   J4
        ori.b   #8,d3
J4:     cmpi.b  #15,d3
        beq.s   +ncol
        cmpi.b  #9,d3
        beq.s   +ncol
        cmpi.b  #6,d3
        beq.s   +ncol
set9:
        move.b  #2,(a2)
        bra.s   +ncol
ch1to0:
        move.l  a3,a2
        move.l  #53760,d4
no5:
        cmpi.b  #1,(a2)+
        bne.s   loop
        clr.b   -1(a2)
loop:
        dbra    d4,no5
ch2to1:
        move.l  a3,a2

        move.l  #53760,d4
no6:
        cmpi.b  #2,(a2)+
        bne.s   loop1
        move.b  #1,-1(a2)
loop1:
        dbra    d4,no6
End:
        move.l  #64,d0
        trap    #15
        .word   2
        move.l  #13,d0
        trap    #15
        .word   2
        move.l  #10,d0
        trap    #15
        .word   2
        movem.l (sp)+,d5/d4/a3
        rts

        .globl  +main
        .globl  +kon
        .globl  +nrow
        .globl  +ncol
```

Figure AD14

```
#include <std.h>
selshad() /*  program to scan image to find pixel and track outline */
/* all Freeman coded lines are storec simultaneously */
{
GLOBAL char mpixel[53761];/* expanded image storage array */
GLOBAL short point[30000];/* Freeman chain coding storage */
GLOBAL short I,J,BP,N;    /* I,J - X,Y coordinates of first point of last segment trackec */
                          /* N - image segment number     */
GLOBAL char *PMPIXEL;     /* pixel pointer to first point of last segment trackec */
GLOBAL short *line[100];  /* array of pointers to starting points of segments ipcint[] */
GLOBAL short *endline[100];/* array of pointers to end points of segments */
GLOBAL short LINE;        /* starting point index */
GLOBAL short bound[100];  /* array of number of segment points */
GLOBAL short *ppoint;     /* chain coding element marker */
register char *pmpixel;   /* image array pointer */
short dir,k;              /* sharp - degree of change of coding direction */
                          /* dir   - chain coding direction */
register short i,j,sharp,v; /* i,j   -x,y coordinates of tracked points */
                          /* k,v   -sweep sequencing variables  */
short a,b,h,p,g,qn;       /* a,h   -starting point coordinates  */
                          /* qn    -tracked pixel marker value  */
                          /* h,p,q -utility variables */
register short *pgenl;    /* general purpose register pointer variable */
BP=0;
qn=N+5;
v=sharp=h=0;
dir=5;
k=(-1);
pmpixel=PMPIXEL;
j=J;
i=I;
line[LINE]=ppoint;
   pgenl=ppoint;
   for(;i<280;i++){ /* to complete to erc of last scanned row */
             if(*(pmpixel++)==1){
               *(pmpixel-1)=qn;
               (*pgenl++)=i;
               (*pgenl++)=j;
               (*pgenl++)=dir;
               (*pgenl++)=0;
               I=i;
               J=j;
               pmpixel--;
               PMPIXEL=pmpixel;
                goto track;
                   }
                            }
         J++;
         j++;
         for(;j<192;j++,J++,pmpixel=pmpixel+280){
/* putfmt("j = %s,J= %s, pmpixel= %i\n",j,J,pmpixel);*/
            for(i=0;i<280;i++){
              if(*(pmpixel+i)==1){
               *(pmpixel+i)=qn;
               (*pgenl++)=i;
               (*pgenl++)=j;
```



```
               (*pgenl++)=dir;
               (*pgenl++)=0;
               a=I=i;
               b=J=j;
               pmpixel=pmpixel+i;
               PMPIXEL=pmpixel;
  /*            putfmt("2nd loop J= %s,i= %s j= %s \n",J,i,j);*/
                goto track;
                        }
                         }
                                                  }
goto halt;
```

```
track:/* main tracking section of program */
        for(;;){
        h++;
           while(dir==5){
                 ++j;
                 pmpixel=pmpixel+280;
                 if (*pmpixel==1){
                    *pmpixel=an;
                    *pgen1++=i;
                    *pgen1++=j;
                    *pgen1++=dir;
                    *(pgen1-4)=sharp;
                    ogen1++;
                    sharp=0;
                    k=(-1);
                    v=0;
                                    }
                 else {
                     sharp++;
                     --j;
                     pmpixel=pmpixel-280;
                 if(sharp>6)
                 goto halt;
                     k=k*(-1);
                     dir=dir+(k*(++v));
                       switch(dir){
                        case 10:
                           dir=2;
                           break;
                        case (-1):
                           dir=7;
                           break;
                                 }
                        }
       }
while(dir==4){
      ++j;
      ++i;
      pmpixel=pmpixel+281;
  if (*pmpixel==1){
     *pmpixel=an;
     *pgen1++=i;
     *pgen1++=j;
     *pgen1++=dir;
```

Figure AD14 Continued.


```
                    *(pgen1-4)=sharp;
                    pgen1++;
                    sharp=0;
                    k=(-1);
                    v=0;
                                    }
                 else {
                    sharp++;
                    --j;
                    --i;
                    pmpixel=pmpixel-281;
                 if(sharp>6)
                 goto halt;
                   k=k*(-1);
                   dir=dir+(k*(++v));
                   switch(dir){
                   case 9:
                     dir=1;
                     break;
                   case (-2):
                     dir=5;
                     break;
                   case 0:
                     dir=8;
                     break;
                                 }
                     }
}
```

```
while(dir==1){
  --j;
  pmpixel=pmpixel-280;
if (*pmpixel==1){
   *pmpixel=gn;
   *pgenl++=i;
   *pgenl++=j;
   *pgenl++=dir;
   *(pgenl-4)=sharp;
   pgenl++;
   sharp=0;
   k=(-1);
   v=0;
                    }
else {
```

Figure AD14 Continued.


```
        sharp++;
        ++j;
       pmpixel=pmpixel+280;
if(sharp>6)
goto halt;
        k=k*(-1);
        dir=dir+(k*(++v));
        switch(dir){
         case (-1):
            dir=7;
            break;
         case -3:
            dir=5;
            break;
         case -5:
            dir=3;
            break;
                  }
           }
     }
while(dir==8){
      --i;
      --j;
  pmpixel=pmpixel-281;
  if (*pmpixel==1){
     *pmpixel=gn;
     *pgenl++=i;
     *pgenl++=j;
     *pgenl++=dir;
     *(pgenl-4)=sharp;
     pgenl++;
     sharp=0;
     k=(-1);
     v=0;
                    }
 else {
   sharp++;
   ++i;
   ++j;
   pmpixel=pmpixel+281;
if(sharp>6)
goto halt;
   k=k*(-1);
   dir=dir+(k*(++v));
   switch(dir){
   case 11:
    dir=7;
    break;
    case 13:
     dir=5;
     break;
    case 9:
     dir=1;
     break;
            }
      }
}
```

```
while(dir==7){
      --i;
      --pmpixel;
  if (*pmpixel==1){
      *pmpixel=on;
      *pgenl++=i;
      *pgenl++=j;
      *pgenl++=dir;
      *(pgenl-4)=sharp;
      pgenl++;
      sharp=0;
      k=(-1);
      v=0;
                       }
 else {
     sharp++;
     ++i;
     ++pmpixel;
    if(sharp>6)
    goto halt;
     k=k*(-1);
     dir=dir+(k*(++v));
       switch(dir){
          case 10:
             dir=2;
             break;
          case 12:
             dir=4;
             break;
                  }
        }
    }
while(dir==6){
      --i;
      ++j;
      pmpixel=pmpixel+279;
  if (*pmpixel==1){
      *pmpixel=on;
      *pgenl++=i;
      *pgenl++=j;
      *pgenl++=dir;
      *(pgenl-4)=sharp;
      pgenl++;
      sharp=0;
      k=(-1);
      v=0;
                       }
 else {
      sharp++;
      ++i;
      --j;
      pmpixel=pmpixel-279;
      if(sharp>6)
      goto halt;
      k=k*(-1);
```

Feb 15 10:17:46 1988 mfrecode.c Page 7

```
      dir=dir+(k*(++v));
      switch(dir){
       case 11:
         dir=3;
         break;
       case 9:
         dir=1;
         break;
                  case 0:
                    dir=8;
                    break;
                         }
                   }
                 }
               }
halt:   /*loop terminated but not necessarily closed*/
        bound[LINE]=pgenl-line[LINE];
        endline[LINE]=(pgenl-4);
        *pgenl++=0xffff;/* endchar */
        LINE++;
        if(J>190){
     /* putfmt("value returned = 0. \n");*/
          ppoint=pgenl;
          return(0);
                 }
        else {
     /*  putfmt("value returned = -1 \n");*/
           ppoint=pgenl;
           return(-1);
          }
}
```

Figure AD14 Continued.

Figure AD15

Feb 15 10:20:11 1988 fclosed.c Page 1

```
#include <sta.h>
fclosed()/* test to cetermine whether a shacow outline is closed */
{
GLOBAL short *line[1CC],*endline[1CC];
GLOBAL short bcund[1CC];
GLOBAL short LINE,lilen;
GLCBAL unsigned short type[100][3];
register unsigned shcrt n,threshx,threshy;
register short *pline,*perdline;
unsigned short closed=0x0010;
threshx=10;
threshy=5;
for(n=0;n<(LINE-1);n++){
   if(bound[n]>lilen){
     oline=line[n];
     perdline=endline[n];
/* putfmt("*line[%s]= %s,*endline[%s]= %s \n",n,*line[n],n,*endline[n]); */
     if((*pline-(*perdline)<threshx)&&(*perdline-(*pline)<threshx)){
   /*      putfmt("Data set %s is half closed.\n",n); */
/* tests proximity of pcints is smaller than threshold */
       if((*(pline+1)-(*(perdline+1))<threshy)&&(*(perdline+1)-(*(pline+1))<threshy)){
       /* labelling of line indexed element */
         type[n][2]=type[n][2]|closed;
    /*     putfmt("Data set %s is closed.\n",n); */
           }
                                                                          }
                    }
                      }
}
```

Feb 15 10:20:11 1988 freperi.c Page 1

Figure AD18

```
#include <std.h>
fperi()/* total perimeter length of chair-cocec lires */
{
GLCPAL short point[3C000],bound[1CC],totperi[1CC],type[1CC][3];
GLCBAL short *line[1CC],*aline[100],*aendline[10C];
GLCBAL short LINE,lilen;
unsigned short n;
register short odds;
register short evens;
register short *pline;
unsigned short endchar=0xffff;
unsigned short closed=0x0010;
evens=odds=0;
for(n=0;n<(LINE-1);n++){
   if((bound[n]>lilen)&&(type[n][2]==closed)){
     for(pline=aline[n],evens=0,odds=0;pline !=(aencline[n]+4);pline +=4){
        if((*(pline+2)==1)||(*(pline+2)==3)||(*(pline+2)==5)||(*(pline+2)==7)){
          odds++;
                                                                          }
         else {
           evens++;
               }
                                                                          }
      totperi[n]=(100*odds+141*evens)/10C;
                                                                          }
                    }
/*       for(n=0;r<(LINE-1);n++){
           if((bourd[n]>lilen)&&(type[n][2]==closed)){
             outfmt("totperi[%s]= %s \n",n,totperi[n]);
                                }
                                  }*/
}
```

Figure AD16

```
#include <std.h>
xend()/* provides artificial ends to chain coded lines so that */
      /* x-coordinate is same for start and finish point */
{
GLOBAL short *line[100],*endline[100],*aline[100],*aendline[100];
GLOBAL short bound[100];
GLOBAL short LINE,lilen;
GLOBAL unsigned short type[100][3];
register short *pline,*penciine;
register unsigned short n,k;
unsigned short m,b;
short bl,bh;
unsigned short closed=0x0010;
bl=215;bh=270;
for(n=0;n<(LINE-1);n++){
   if((bound[n]>bl)&&(bound[n]<bh)&&(type[n][2]==closed)){ /* protect the calibration circle. */
       aline[n]=line[n];aendline[n]=endline[n];
     /*  putfmt("Line %s x-adjusted case 0.\n",n); */
                                                        }
   else {
    if((bound[n]>lilen)&&(type[n][2]==closed)){
      pline=line[n];
      perdline=endline[n];
/* test only valid if particular line is above the threshold line length. */
     if((*pline)==(*(perdline))){
       aline[n]=pline;aendline[n]=perdline;
    /*   putfmt("Line %s x-adjusted case 1.\n",n); */
                                  }
     else { /* to ensure that endpoints are not adjusted if */
            /* inside edge first slopes right to left       */
          b=0;
          if(*(pline)<(*(pencline))){
            if(*(pline+4)<=(*(pline))){
              if(*(pline+8)<=(*(pline+4))){
                if(*(pline+12)<=(*(plire+8))){
                  if(*(pline+16)<=(*(pline+12))){
                    aline[n]=pline;aencline[n]=percline;
                    b=1;
/*                  putfmt("Line %s x-adjusted case 2.\n",n); */
                                                  }
                                                }
                                              }
                                            }
                                          }
      if(b!=1){
        if(*(pline)>(*(pendline))){
        for(k=0,b=0;k<bound[n]&&(b !=1);k +=4){
           if(*(pencline-k)==(*pline)){
             aline[n]=pline;aendline[n]=(percline-k);b=1;
/*             putfmt("Line %s x-adjusted case 3.k= %s.\n",n,k); */
                                                }
                                              }
                                            }
       else {
         for(k=4,b=0;k<bound[n]&&(b !=1);k +=4){
           if(*(pline+k)==(*pendline)){
```



```
                  aline[n]=pline+k;aencline[n]=percline;b=1;
          /*        putfmt("Line %s x-adjusted case 4. k= %s.\n",n,k); */
                                                  }
                                                }
                                              }
    }
                                                }
  }
          }
}
```

Figure AD17


```
#include <std.h>
farea()/* area of chain-coded closed shadow */
{
GLOBAL short point[30000],bound[100],totarea[100],type[100][3];
GLOBAL short *line[100],*endline[100],*aline[100],*aendline[100];
GLOBAL short LINE,lilen;
register short area;
register unsignec short *pline;
register short j,dir;
unsigned short n;
unsigned short endchar=0xffff;
unsigned short closed=0x0010;
for(n=0;n<(LINE-1);n++){
   if((bound[n]>lilen)&&(type[n][2]==closed)){
     for(pline=(aline[n]+4),area=0;pline !=(aendline[n]+4);pline +=4){
        dir=(*(pline+2));
          j=(*(pline+1));
          j <<=1;/* multiply each column element by two */
        switch(dir){
           case 1:
                  break;
           case 5:
                  break;
           case 2:
                  j++;
                  area=area+j;
                  break;
           case 3:
                  area=area+j;
                  break;
           case 4:
                  j--;
                  area=area+j;
                  break;
           case 6:
                  j--;
                  area=area-j;
                  break;
           case 7:
                  area=area-j;
                  break;
           case 8:
                  j++;
                  area=area-j;
                  break;
                                  }
        /*   putfmt("a= %s, ",area); */
                                 }
        area >>=1;/* divide resultant area by two */
/*      putfmt("tarea= %s. \n",area); */
        totarea[n]=area;
                             }
                           }
/* for(n=0;n<(LINE-1);n++){
     if((bound[n]>lilen)&&(type[n][2]==closed)){
       putfmt("totarea[%s]= %s \n",n,totarea[n]);
```



```
                                  }
                                  }*/
}
```

PAGE NUMBERING AS ORIGINAL

```
#include <std.h>
compact()/* compaction factor calculator */
{
GLOBAL short LINE,lilen;
GLOBAL short bound[100],totperi[100],totarea[100],paction[100],type[100][3];
register short *ptotperi,*ptotarea,*ppaction;
register short n;
register unsigned short closed=0x0010;
short l;
ptotperi=totperi;
ptotarea=totarea;
ppaction=paction;
l=(LINE-1);
        for(n=0;n<l;n++,ptotperi++,ptotarea++,ppaction++){
           if((bound[n]>lilen)&&(type[n][2]==closed)){
             *ppaction=(*ptotperi*(*ptotperi))/(*ptotarea);
                              }
                                }
/* for(n=0;n<(LINE-1);n++){
   if((bound[n]>lilen)&&(type[n][2]==closed)){
      putfmt("paction[%s]= %s \n",n,paction[n]);
                    }
                      }*/
}
```

Figure AD19

Figure AD20

```
#include <std.h>
shadw() /*to determine min-y max-y max-x ensuring inner edge & finding line lengths */
{
        GLOBAL short point[30000],impoint[100][10],bound[100],paction[100];
        GLOBAL short *line[100];
        GLOBAL unsigned short type[100][3];
        GLOBAL short LINE,lilen,impl;
        int s,t;
        short imax,jmax,endchar,q,minpid,maxpid,i,j,y,xend,greater,less,bl,bh;
        unsigned short closed;
        register short pid,apid;
        register short *ppoint=point;
        register short *pimpoint=impoint;
        register short *appoint;
        endchar=0xffff;
        closed=0x0010;
        impl=0;
        bl=200;
        bh=280;
        for(s=0;s<(LINE-1);s++){
        if((bound[s]>lilen)&&((type[s][2]&closed)==closed)){
        if((bound[s]<bl)||(bound[s]>bh)){
        /* Exclude feature derivation on test circle. */
        /* putfmt("Feature point selection on boundary %i.\n",s); */
        ppoint=line[s]+80;/* Start testing 20 points along line. */
        pimpoint=(&impoint[s][0]);
        minpid=maxpid=0;
        /* ROUTINE TO WHICH EDGE OF SHADOW IS TRACKED FIRST. */
        pid=(*(ppoint+3));
        y=(*(ppoint+1));
        for(;(*ppoint != endchar) && (*(ppoint+1)<(y+5));ppoint=ppoint+4,pid++){
                appoint=(ppoint+60); /* To ensure no flat section spoils */
                apid=(*(appoint+3));
                for(j=(*(ppoint+1)),i=(*ppoint);*appoint != endchar;appoint=appoint+4,apid++){
                        if((*(appoint+1)==j) && (*(appoint)<i) && (*(appoint-3)>j)){
                                goto reverse;
                        }
                }
        }
        /* ROUTINE TO FIND STARTING POINT - 1st point x increasing */
        for(pid=0,q=0,ppoint=line[s];(*(ppoint+4)!= endchar)&&(q !=(-1));ppoint=ppoint+4,pid++){
        /* putfmt("*(ppoint+4)= %s,*ppoint = %s,*ppoint+5 = %s,*ppoint+1= %s,pid= %s \n"
        ,*(ppoint+4),*ppoint,*(ppoint+5),*(ppoint+1),pid); */
                if((*(ppoint+4)>*ppoint) && (*(ppoint+5)>(*(ppoint+1)))){
                if((*(ppoint+8)>=*(ppoint+4)) && (*(ppoint+9)>=(*(ppoint+5)))){
                if((*(ppoint+12)>=*(ppoint+8)) && (*(ppoint+13)>=(*(ppoint+9)))){
                if((*(ppoint+16)>=*(ppoint+12)) && (*(ppoint+17)>=(*(ppoint+13)))){
                if((*(ppoint+20)>=*(ppoint+16)) && (*(ppoint+21)>=(*(ppoint+17)))){
                if((*(ppoint+24)>=*(ppoint+20)) && (*(ppoint+25)>=(*(ppoint+21)))){
                                *(pimpoint)=(*ppoint);
                                *(pimpoint+1)=(*(ppoint+1));
                                *(pimpoint+2)=minpid=pid;
                                q=(-1);
                                                                        }
                                                                        }
                                                                        }
```

Figure AD20 Continued.

```
                                                                    )
                                                                  )
            }
        }
        /* ROUTINE TO FIND MAX POINT - max y with x decrasing */
        if(q==(-1)){/* to ensure if 1st point not set then neither are other two */
          *(pimpoint+6)=(*(pimpoint+3))=(*(pimpoint));/*dflt set of max when min set.*/

          *(pimpoint+7)=(*(pimpoint+4))=(*(pimpoint+1));
          *(pimpoint+8)=(*(pimpoint+5))=maxpid=(*(pimpoint+2));
          q=0;
          pid=minpid;
          jmax=0;
          ppoint=(line[s]+4*minpid);
            for(;*(ppcint+4)!=endchar && q!=(-1);ppoint=ppoint+4,pid++){
        /* putfmt("*(ppoint+4)= %s,*ppoint = %s,*ppcint-3 = %s,*ppoint+1= %s,pid= %s ,jmax= %s\n"
        ,*(ppoint+4),*ppcint,*(ppoint-3),*(ppoint+1),pid,jmax); */
            if((*(ppcint+1)>=jmax)&&(*(ppoint+4)<(*ppoint))){
            if(*(ppoint+8)<=(*(ppoint+4))){
            if((*(ppcint)<=(*(ppoint-4))) && (*(ppoint+1)>=(*(ppoint-3)))){
            if((*(ppcint-4)<=(*(ppcint-8))) && (*(ppcint-3)>=(*(ppoint-7)))){
            if((*(ppoint-8)<=(*(ppoint-12)))&& (*(ppcint-7)>=(*(ppoint-11)))){
            if((*(ppcint-12)<=(*(ppoint-16)))&&(*(ppcint-11)>=(*(ppoint-15)))){
                        jmax=(*(ppoint+1));
                        *(pimpoint+6)=(*ppcint);
                        *(pimpoint+7)=(*(ppcint+1));
                        *(pimpoint+8)=maxpid=pid;
            if((*(ppcint+8)<=(*(ppoint+12)))&&(*(ppcint+12)<=(*(ppoint+16)))){
            if((*(ppcint+16)<=(*(ppoint+20)))&&(*(ppcint+20)<=(*(ppoint+24)))){
            if(*(ppoint+30)>(*(ppoint+4))){ /* 20 points along x greater */
              q=(-1);
                                        )
                                                                          }
                                                                        )
                                                                    }
                                                                    }
                                                                    }
                                                                    }
                                                                    }
                                                                    }
                                                                    }
                    }
                    }
            /* ROUTINE TO FIND MIDPOINT - max x between min and max */
            for(imax=0,pid=minpid,ppoint=(line[s]+4*minpid);pid <maxpid;ppoint=ppcint+4,pid++){
                    if(((*ppoint)>imax)&&(*(ppcint+4)>=(*ppcint))&&(*(ppoint+8)>=(*(ppcint+4)))){
                            imax=(*(pimpcint+3))=(*ppoint);
                            *(pimpoint+4)=(*(ppcint+1));
                            *(pimpoint+5)=pid;
                    }
            }
            continue;/* execute next value in main loop */
    reverse:
            /* ROUTINE TO FIND STARTING POINT BACKWARDS -x decreasing y-min for*/
            putfmt("Reverse edge point calculation routine entered.\n");
            for(q=0,pid=(apid-20),ppoint=(appoint-80);*(ppoint+4)!=endchar;ppoint=ppcint+4,pid++){
               if((*(ppoint-4)>*ppoint) && (*(ppoint-3)>*(ppcint+1))){
```



```
                if((*(ppoint-8)>=(*(ppoint-4))) && (*(ppcint-7)>=(*(ppcint-3)))){
                if((*(ppoint-12)>=(*(ppoint-8))) && (*(ppcint-11)>=(*(ppoint-7)))){
                if((*(ppoint-16)>=(*(ppoint-12))) && (*(ppoint-15)>=(*(ppoint-11)))){
                if((*(ppoint-20)>=(*(ppoint-16))) && (*(ppoint-19)>=(*(ppoint-15)))){
                if((*(ppoint-24)>=(*(ppoint-20))) && (*(ppoint-23)>=(*(ppoint-19)))){
                            *pimpoint=(*ppcint);
                            *(pimpoint+1)=(*(ppcint+1));
                            *(pimpoint+2)=minpid=pid;/* larger than maxpid */
        /*          putfmt("s= %i, impoint[s][0]= %s, impcint[s]= %s, impoint[s]=%s.\n"
        ,s,impoint[s][0],impcint[s][1],impoint[s][2]); */
                                                                        )
                                                                        )
                                                                        )
                                                                    }
                                                                }
                                                            }
                                                        }
                    }
```

```
        /* ROUTINE TO FIND MAX POINT BACKWARDS */
        for(q=0,pid=apid,jmax=0,ppoint=appoint;(pid >=0) && (q != (-1));ppcint=ppcint-4,pid--){
           if((*(ppoint+1)>=jmax)&&(*(ppcint-4)<(*ppcint))){
           if(*(ppcint-8)<= (*(ppoint-4))){
           if((*(ppoint)<=(*(ppoint+4))) && (*(ppoint+1)>=(*(ppoint+5)))){
           if((*(ppcint+4)<=(*(ppoint+8))) && (*(ppcint+5)>=(*(ppcint+9)))){
           if((*(ppoint+8)<=(*(ppoint+12))) && (*(ppoint+9)>=(*(ppoint+13)))){
           if((*(ppcint+12)<=(*(ppoint+16))) && (*(ppcint+13)>=(*(ppoint+17)))){
                jmax=(*(ppcint+1));
                *(pimpoint+6)=(*ppoint);
                *(pimpcint+7)=(*(ppcint+1));
                *(pimpoint+8)=maxpid=pid;
/*  putfmt("s= %i, impoint[s][6]= %s, impoint[s]= %s, impcint[s]=%s.\n"
,s,impoint[s][6],impcint[s][7],impoint[s][8]); */
            if((*(ppcint-8)<=(*(ppoint-12)))&&(*(ppcint-12)<=(*(ppoint-16)))){
            if((*(ppcint-16)<=(*(ppoint-20)))&&(*(ppoint-20)<=(*(ppoint-24)))){
            if(*(ppoint-80)>(*(ppoint-4))){ /* 20 points along x greater */
              q=(-1);
                                        }
                                                                          }
                                                                        }
                                                                      }
                                                                    }
                                                                  }
                                                                }
                                                              }
        /* ROUTINE TO FIND MIDPOINT */
        for(pid=apid,imax=0,ppoint=apcpcint;pid >=maxpid;ppoint=ppcint-4,pid--){
                if((*ppoint>imax)&&(*(ppcint-4)>=imax)&&(*(ppoint-8)>=imax)){
                        imax=(*(pimpcint+1))=(*ppoint);
                        *(pimpoint+4)=(*(ppcint+1));
                        *(pimpoint+5)=pid;
 /* putfmt("s= %i, impoint[s][3]= %s, impcint[s]= %s, impcint[s]=%s.\n"
,s,impoint[s][3],impcint[s][4],impoint[s][5]); */
                }
        }
        impl++;
```

Figure AD20 Continued.


```
     }
    }
   }
}
```

Figure AD21


```
#include<std.h>
char cimp()/* production of features from important points */
{
GLOBAL short impcirt[100][10],cimpoint[100][25];
GLOBAL short LINE;
DOUBLE sqrt();
char getch();
register short *pimpcint=impoint;
register short q,t;
double d,f,g;
short l;
l=(LINE-1);
for(q=0;q<l;q++){
   pimpoint=(&impcint[q][0]);
   for(t=0;t<9;t++){
      cimpoint[q][t]=(*(pimpoint+t));
                   }
      cimpoint[q][9]=(*(pimpcint+3)-(*pimpcint));/* mindifx */
      cimpcint[q][10]=(*(pimpoint+3)-(*(impcint+6)));/* maxdifx */
      cimpoint[q][11]=(*(pimpoint+6)-(*pimpcint));/* maxx-minx */
      cimpoint[q][12]=(*(pimpcint+4)-(*(impcint+1)));/* midy-miny */
      cimpoint[q][13]=(*(pimpoint+7)-(*(impcint+4)));/* maxy-midy */
      cimpcint[q][14]=(*(pimpcint+7)-(*(pimpcirt+1)));/* maxy-miny */
      f=(cimpcint[q][9])*(cimpcint[q][9])+(cimpcint[c][12])*(cimpcint[q][12]);
      cimpoint[q][15]=(short)(sqrt(f));
      g=(cimpcint[q][10])*(cimpcint[q][10])+(cimpcint[q][13])*(cimpoint[c][13]);
      cimpoint[q][16]=(short)(sqrt(g));
      d=(cimpoint[q][11])*(cimpoint[q][11])+(cimpcint[q][14])*(cimpoint[c][14]);
      cimpoint[q][17]=(short)(sqrt(d));
      if(cimpcint[q][9]>cimpoint[q][10]){
        cimpcint[q][18]=cimpoint[q][10];
                                       }
       else cimpcint[q][18]=cimpoint[q][9];
       cimpoint[q][19]=cimpoint[q][15]+cimpcint[q][16];
                         }
}
```

```
#include<std.h>
ccircle()
{
GLOBAL short point[3COCO],bound[1CC],totarea[1CC],cimpcint[1CO][2^];
GLOBAL short LINE,llen,*line[100],totperi[100];
int s,t,r;
register short *ppcint;
short minx,maxx,miry,maxy,cenx,ceny;
short bl,bh,ph,pl,ah,al,endchar;
encchar=0xffff;
minx=miny=20C;
maxx=maxy=0;
ph=77;pl=63;
ah=5CO;al=395;
bl=22C;
bh=26^;
for(s=0;s<(LINE-1);s++){
   if((bound[s]>bl)&&(bound[s]<bh)&&(totperi[s]>pl)&&(totperi[s]<ph)&&(totarea[s]>al)&&(totarea[s]<ah)){
       for(ppcint=line[s];*ppoint !=encchar;ppcint +=4){
          if(*ppcint<minx){
            minx= *ppcint;
                          }
          if(*ppcint>maxx){
            maxx= *ppoint;
                          }
          if(*(ppcint+1)<miny){
            miny= *(ppcint+1);
                          }
          if(*(ppcint+1)>maxy){
            maxy= *(ppcint+1);
                          }
                                                  }
      cimpoint[s][20]=cenx=(minx+maxx)/2;
      cimpoint[s][21]=ceny=(miny+maxy)/2;
      goto end;
                                }
                       }
end:
    putfmt("Centre marker centre calculatec. x= %s, y= %s.\n",cenx,ceny);
}
```

Figure AD22


```
#include<std.h>
dcentre() /* Routine to calculate distance from object of interest */
          /* to pick-up marker centre.                             */
{
GLOBAL short bound[100],totperi[10C],cimpcint[1CC][25];
GLOBAL short LINE,llen,*line[10C],totarea[100];
DOUBLE sqrt();
int s,n;
register short *pncint;
short mind,midd,maxd;
int minds,maxds,midds;
short bl,bh,al,ah,pl,ph,sigar,sigtot;
sigar=400;
sigtot=20;
ph=77;pl=63;
ah=500;al=395;
bl=20C;
bh=265;
for(s=0;s<(LINE-1);s++){
   if((bound[s]>llen)&&(totarea[s]>sigar)&&(cimpcint[s][19]>sigtot)){
     for(n=s+1;n<(LINE-1);n++){
        if((bound[n]>bl)&&(bound[n]<bh)&&(totperi[n]>pl)&&(totperi[n]<ph)&&(tctarea[n]>al)&&(tctarea[n]<ah)
          putfmt("Calibration circle found. n= %s.\r",n);
            minds=((cimpcint[s][0]-cimpcint[n][2C])*
                   (cimpcint[s][0]-cimpcint[n][2C]))+
                  ((cimpcint[s][1]-cimpcint[n][21])*
                   (cimpcint[s][1]-cimpoint[n][21]));
            midds=((cimpoint[s][3]-cimpcint[n][20])*
                   (cimpcint[s][3]-cimpcint[n][20]))+
                  ((cimpcint[s][4]-cimpcint[n][21])*
                   (cimpcint[s][4]-cimpcint[n][21]));
            maxds=((cimpcint[s][6]-cimpcint[n][2C])*
                   (cimpcint[s][6]-cimpcint[n][20]))+
                  ((cimpcint[s][7]-cimpcint[n][21])*
                   (cimpcint[s][7]-cimpcint[n][21]));
putfmt("minds=%i, midds=%i, maxds=%i.\n",minds,midds,maxds);
      cimpoint[s][20]=cimpcint[n][20];
      cimpoint[s][21]=cimpcint[n][21];
      cimpoint[s][22]=(short)(sqrt((double)(minds)));
      cimpcint[s][23]=(short)(sqrt((double)(midds)));
      cimpoint[s][24]=(short)(sqrt((double)(maxds)));
      goto end;
                                }
                                }
                       }
                       }
end:
putfmt("\n");
/* outfmt("Distances to centre marker at %s, %s are:- ",cimpcint[s][20],cimpcint[s][21]);
putfmt("mind= %i, midd= %i, maxd= %i.\n",cimpoint[s][22],cimpoint[s][23],cimpcint[s][24]); */
```

Figure AD23

Figure AD24

```
#include <std.h>
timp()
{
GLOBAL short timpcint[500][15];
GLOBAL short totperi[100];
GLOBAL short totarea[100];
GLOBAL short paction[100];
GLOBAL short bound[100];
GLOBAL short cimpoint[100][25];
GLOBAL short lilen;
GLOBAL int LS;
GLOBAL short LINE;
GLOBAL short FI;
register short q,t,set;
short m=0;
FI++;
for(q=0;q<(LINE-1);q++){
   if((bound[q]>lilen)&&(paction[q]>15)&&(cimpcint[q][19]>10)){
      m=1;
      timpcint[LS][0]=FI;/* Frame index */
      for(t=0,set=0;t<9;t +=3){ /* Testing for invalid feature sets.*/
         if((cimpoint[q][t]==0)||(cimpcirt[q][t+1]==0))set=1;
                               }
      if(totperi[q]==0)set=1;
      if(set==1){ /* If bad data set make all entries zero.*/
        for(t=1;t<12;t++){
           timpoint[LS][t]=0;
                         }
              }
      else {
           timpoint[LS][1]=totperi[q];
           if(totarea[q]<0){
             timpoint[LS][2]=(-totarea[q]);
             timpcint[LS][3]=(-paction[q]);
                           }
           else {
             timpcint[LS][2]=totarea[q];
             timpoint[LS][3]=paction[q];
                }
             timpcint[LS][4]=cimpoint[c][19];
             timpcint[LS][5]=cimpoint[q][17];
             timpcint[LS][6]=cimpoint[q][18];
             timpoint[LS][7]=cimpoint[q][15];
             timpcint[LS][8]=cimpoint[c][16];
             timpcint[LS][9]=cimpcint[c][22];
             timpoint[LS][10]=cimpcint[q][23];
             timpoint[LS][11]=cimpoint[q][24];
          }
          LS++;
      putfmt("Values being stored for output are:- %s,%s,%s,%s,%s,%s,%s,"
,FI,totperi[q],totarea[q],paction[q],cimpcint[c][19],cimpoint[q][17],cimpcint[c][18]);
      putfmt("%s,%s,%s,%s,%s.\n"
,cimpcint[q][15],cimpoint[q][16],cimpcint[q][22],cimpoint[q][23],cimpoint[q][24]);
                       }
                        }
      if(m!=1){ /* If all bad data set make zero data set.*/
```



```
         timpoint[LS][0]=FI;/* Frame index */
         for(t=1;t<12;t++){
            timpcint[LS][t]=0;
                          }
            LS++;
      putfmt("Values being stored for cutout are:- %s,%s,%s,%s,%s,%s,%s,"
,FI,totperi[q],totarea[q],paction[q],cimpcint[q][19],cimpoint[q][17],cimpcint[q][18]);
      putfmt("%s,%s,%s,%s,%s.\n"
,cimpoint[q][15],cimpoint[q][16],cimpcint[q][22],cimpoint[q][23],cimpoint[q][24]);
               }
}
```

Figure AD25

```
#include <std.h>
struct FILEDATA
        {
        char name[16];
        int block;
        int load;
        int start;
        int count;
        char *pbuf;
        int fsize;
        };
toffile()
{
extern short timpoint[500][15];
extern struct FILEDATA fdata,*fss;
extern int LS;
short buf[7000];
short nn,q,t;
char fname[16];
char *pfname;
int n,drive,ip;
fss = &fdata;
drive=1;
pframe=fname;
putfmt("Feature data storage routine entered.\r\n");
putfmt("Place formatted feature data storage disc in drive 1.\n\n");
putfmt("Enter the name of the file in which the feature data is to be stored.\n");
getfmt("%p",pframe);
if((n=system(7,&drive))!=0){
  putfmt("Error in accessing drive 1.\n");
  putfmt("Value of n returned is %i.\n",n);
                          }
if((n=system(16))!=0){
  putfmt("Error in restoring drive 1.\n");
  putfmt("Value of n returned is %i.\n",n);
                       }
for(n=0;n<15;n++){ /* Leave the null termination */
   fdata.name[n]=frame[n];
                  }
if((n=system(9,fss))!=0){
  putfmt("Error in opening the file.\n");
  putfmt("Value of n returned is %i.\n",n);
  putfmt("Disk storage program aborts.\r");
  goto end;
                        }
for(nn=0,q=1;nn<LS;nn++){
   for(t=0;t<12;t++){
      buf[q++]=timpoint[nn][t];
                     }
                    }
buf[0]=q-1; /* Total number of sets of data to be o/p */
/* Number to be o/p =q/256 +1 to make up fractional part.*/
putfmt("Number of buffer elements= %s, q/256= %s.\n",q,q/512+1);
fdata.pbuf=buf;
fdata.count=q/256+1; /* Write out q/256+1 blocks */
      if((n=system(14,fss))!=0){
```



```
        putfmt("Error in writing to disc file %p.\n",fdata.name);
        putfmt("Value of n returned is %i.\r",n);
        putfmt("block= %i, count= %i.\n",fdata.block,fdata.count);
                              }
if((n=system(11,fss))!=0){
  putfmt("Error in closing file.\n");
  putfmt("Value of n returned is %i.\n",n);
                         }
end:
system(8);
putfmt("Drive 1 deselected.\n");
}
```

Figure AD26

```
#define startchar 0x11
#define endchar 0x13
#include <std.h>
toduet()/* Program tc tranmit a run of image features */
        /* for storage on the DUET.                    */
{
GLCBAL short timpoint[500][15];
GLC9AL int LS;
int q,t,w;
short buf[15];
putfmt("Data trarsmission routine enterec.\r\n");
putfmt("Press Carriage return to continue.\n\n");
getch():getch():
buf[0]=startchar;
hwrite3(&buf[0],2);/* Write out the start character */
buf[0]=LS;/* Buffer type-conversion puts value in two bytes */
putfmt("Number of cata sets to be output = %i \n",LS);
hwrite3(&buf,2);/* Output number of sets of cata to be read */
for(q=0;q<LS;q++){
   for(t=0;t<=11;t++){
   buf[t]=timpoint[q][t];
   if((buf[t] & 0xff)==endchar){
     buf[t]++;
                              }
                    }
 putfmt("Values being written out are :- %s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s.\n"
,buf[0],buf[1],buf[2],buf[3],buf[4],buf[5],buf[6],buf[7],buf[8],buf[9],buf[10],buf[11]);
   for(w=0;w<5CC00;w++){  /* Wait locp */ }
   hwrite3(&buf,24);
                    }
buf[0]=endchar;
hwrite3(&buf,2);
}
```

Figure AD27

```
#include<std.h> /* include standard whitesmith's header. */
#include"mylib.h"
/* Declaration and initialization of external variables. */
struct clusinfo cl=0;
struct FILEDATA fdata= {
        "++++++++++++++++",
        0,0,0,15,NULL,0
                        };
struct FILEDATA *fss=NULL;
short nsets=0;
short nclus=1;
short mnclus=0;
short nrange=0;
short crange=0;
short mnrange=0;
float mtot[12]=0;
float var[30][12]=0;
short maxDtoM[3][30]=0;
short minDtoM[3][30]=0;
short cenD[3][30]=0;
short maxvar[30][12]=0;
short minvar[30][12]=0;
short npoints[30]=0;
short ptlist[30][400]=0;
int invarmax=0;
int fvarmax=0;
short PRINTSET=0;
main() /* Coordinating function for cluster analysis.              */
       /* Reads from feature-datafile stores in cluster-data file. */
{
GLOBAL short PRINTSET;
short crit2().crit1();
PRINTSET=0;
cache();          /* Set cache running.*/
initdu();         /* Initialize on-board UART */
       greet();
       cluscrit();
       fromfile();
       putfmt("All specified data read in from disc.\n\n");
       standardize2();
       putfmt("Stardardization of data complete.\n\n");
       while(crit1()!=(-1)){
          meansdnp2();
          putfmt("The mean and variance of each cluster has now been calculated.\n");
          varmaxt1();
          putfmt("Test finished to determine cluster containing the feature with the max variance.\n"
          clusplit();
          putfmt("The cluster with the feature with the largest variance has now been split.\n");
       /* clusprint(); */
                          }
       putfmt("The initial number of clusters has been reached.\n");
       meansdnp2();
       clusprint();
       putfmt("About to enter movement stage.\n");
       move2();
       putfmt("Movement of points to nearest cluster centre completed.\n");

       clusprint();
       putfmt("Printout of initial cluster groupings complete.\n");
       putfmt("About to enter range testing loop for the first time.\n");
       while(crit2()!=(-1)){
            meansdnp2();
            varmaxt2();
            putfmt("Test finished to find cluster with maximum range in distance feature.\n");
            clusplit();
            putfmt("The cluster with the largest range has been split.\n");
                           }
       clusprint();
       putfmt("Printout of distance cluster groupings complete.\n");
       move2();
       clusprint();
       putfmt("About to enter range testing loop for the second time.\n");
       while(crit2()!=(-1)){
            meansdnp2();
            varmaxt2();
            putfmt("Test finished to find cluster with maximum range in distance feature.\n");
            clusplit();
            putfmt("The cluster with the largest range has been split.\n");
                           }
       clusprint();
       move2();
       clusprint();
       putfmt("Printout of final cluster groupings complete.\n");
       detinfo();
       putfmt("Information to be placed in storage file has been determined.\n");
       storfile();
       putfmt("Storage sequence for cluster information completed.\n");

exit();
}
```

Figure AD29

```
#include<std.h>
#include"mylib.h"
fromfile() /* Reads in specified feature data file from disc */
{
extern struct clusinfo cl;
extern short nsets;
extern struct FILEDATA fdata,*fss;
char fname[16];
float *pfeat;
short buf[5000],nob,tbr,q,lessets;
int n,drive,k,t1,t2;
char *pfname;
fss = &fdata;
pframe=fname;
drive=1;
putfmt("Enter the name of the FEATURE data file to be read in.\n");
getfmt("%p",pframe);
putfmt("Enter the set index of the final feature data to be read in.\n");
getfmt("%s",&nsets);
putfmt("Data sets with index up to %s are to be read in from file \"%p\".\n",nsets,pframe);
if((n=system(7,&drive))!=0){
  putfmt("Error in accessing drive 0.\n");
  goto end;
                           }
if((n=system(16))!=0){
  putfmt("Error in restoring drive 0.\n");
  goto end;
                        }
for(n=0;n<15;n++){ /* Leave the null termination. */
   fdata.name[n]=fname[n];
                  }
if((n=system(10,fss))!=0){
  putfmt("Error in opening file for read.\n");
  goto end;
                          }
fdata.pbuf=buf;
t1=(int)(fdata.pbuf);
fdata.count=1;
if((n=system(13,fss))!=0){
  putfmt("Error in reading file.\n");
  goto end;
                          }
nob=buf[0]*2;
if(nob<=510)fdata.count=0;
else {fdata.count=(nob+2)/512);
fdata.pbuf +=512;
t1=(int)(fdata.pbuf);
/* putfmt("fdata-buf=%hi, buf[0]= %s,nob= %s,fdata.count=%i.\n",t1,buf[0],nob,fdata.count); */
if((r=system(13,fss))!=0){
  putfmt("Error in reading file.\n");
  goto end;
                          }
t1=(int)(fdata.pbuf);
/* putfmt("fdata-buf= %hi.\n",t1); */
n=1;
for(pfeat=cl.featvars[0],q=1;(buf[q]!=(nsets+1))&&((buf[q]==(buf[q-12]+1))
```



```
||(buf[q]==(buf[q-12]))||(buf[q]==1));pfeat +=12,n++){
   while(buf[q+1]==0){ /* Ignore data */
     q +=12;
                      }
   *(pfeat+6)=(float)buf[q++];
   *(pfeat+4)=(float)buf[q++];
   *(pfeat+5)=(float)buf[q++];
   *(pfeat+0)=(float)buf[q++];
   *(pfeat+7)=(float)buf[q++];
   *(pfeat+1)=(float)buf[q++];
   *(pfeat+8)=(float)buf[q++];
   *(pfeat+2)=(float)buf[q++];
   *(pfeat+3)=(float)buf[q++];
   *(pfeat+9)=(float)buf[q++];
   *(pfeat+10)=(float)buf[q++];
   *(pfeat+11)=(float)buf[q++];
putfmt("Values for data set %s are: %f, %f, %f, %f, %f, %f, %f, %f, %f, %f.\n",n,*(pfeat+6)
,*pfeat,*(pfeat+1),*(pfeat+2),*(pfeat+3),*(pfeat+4),*(pfeat+5),*(pfeat+9),*(pfeat+10),*(pfeat+11));
                                                    }
nsets=(n-1); /* actual number of sets of data. */
end:
system(9);
putfmt("Drive 1 deselected.\n");
}
```

Feb 16 13:33:05 1988 cluscrit.c Page 1

```
#include<std.h>
cluscrit() /* Routine to input external clustering criteria. */
{
GLOBAL short mnclus,mnrange,PRINTSET;
short accur;
putfmt("Cluster detail entry.\n\n");
putfmt("Do you want to print out the clustering secuence?\n");
if(getch()=='y'){
  PRINTSET=1;
  greet();
  putfmt("The clustering sequence will be sent to the printer.\n");
                }
getch();getch();
putfmt("How many clusters do you want to divide your data into initially.\n");
getfmt("%s",&mnclus);
putfmt("Enter the accuracy required in the robot pick-up point in mm.\n");
getfmt("%s",&accur);
mnrange=accur*3/2; /* *3/4 for pixels, *2 for range */
putfmt("The number of clusters the data is to be divided into initially is :-%s.\n",mnclus);
putfmt("The accuracy in the pick-up point required is %s mm.\n",accur);
putfmt("This corresponds to a %s pixel range.\n",mnrange);
}
```

Figure AD28

Feb 16 13:33:05 1988 standardize2.c Page 1

```
#include<std.h>
#include"mylib.h"
standardize2() /* Function to standardize all input feature data */
{
GLOBAL struct clusinfo cl;
GLOBAL short nsets;
GLOBAL float mtot[12];
double sqrt();
float tot[12],totdsvars[12];
double sig[12];
int m,n,iloop;
for(iloop=0;iloop<12;iloop++){
   tot[iloop]=totdsvars[iloop]=sig[iloop]=0;
                              }
    for(n=0;n<nsets;n++){
       for(iloop=0;iloop<12;iloop++){
          tot[iloop]=tot[iloop]+cl.featvars[n][iloop];
                                   }
                         }
/*    putfmt("Total for variables are:- %f,%f,%f,%f,%f,%f,%f,%f,%f .\n"
,tot[0],tot[1],tot[2],tot[3],tot[4],tot[5],tot[6],tot[7],tot[8]);*/
    for(iloop=0;iloop<12;iloop++){
       mtot[iloop]=tot[iloop]/(float)nsets;
                                 }
    putfmt("Averages of totals of variables are:- %f,%f,%f,%f,%f,%f,%f,%f,%f.\n"
,mtot[0],mtot[1],mtot[2],mtot[3],mtot[4],mtot[5],mtot[6],mtot[7],mtot[8]);
    for(n=0;n<nsets;n++){
       for(iloop=0;iloop<12;iloop++){
          cl.dsfeatvars[n][iloop]=(cl.featvars[n][iloop]-mtot[iloop])*
                                  (cl.featvars[n][iloop]-mtot[iloop]);
          totdsvars[iloop]=totdsvars[iloop]+cl.dsfeatvars[n][iloop];
/*          putfmt("Delta squared = %f , Total of delta squared = %f .\n"
,cl.dsfeatvars[n][iloop],totdsvars[iloop]); */
                                  }
                         }
     for(iloop=0;iloop<12;iloop++){
        /* putfmt("Square root is %d.\n",sqrt((double)totdsvars[iloop])); */
        sig[iloop]= sqrt((double)totdsvars[iloop]/(double)nsets);
        putfmt("The value of sigma for variable %i is %4.2f \n",iloop,sig[iloop]);
                                  }
     for(n=0;n<nsets;n++){
/*      putfmt("The standardized values for the %i th. data set are:-\n",n);
        putfmt("For feature "); */
        for(iloop=0;iloop<12;iloop++){
           cl.sfeatvars[n][iloop]=(cl.featvars[n][iloop]-mtot[iloop])/
                                  (float)sig[iloop];
           /* putfmt("%i = %.2f, ",iloop,cl.sfeatvars[n][iloop]); */
                                  }
           /* putfmt("\n"); */
                         }
}
```

Figure AD30

Feb 16 13:33:05 1988 critl.c Page 1

```
#include<std.h>
critl() /* First clustering criterion checher */
        /* Have the required number of initial clusters been generated? */
{
GLOBAL short nclus;
GLOBAL short mnclus;
if(nclus>=mnclus){
  return(-1);
                 }
else return(0);
}
```

Figure AD31

Figure AD32

```
Feb 16 13:36:06 1988  meansdnp2.c Page 1


#include<std.h>
#include"mylib.h"
meansdnp2() /* Calculate the mean and sd of all existing clusters. */
{
GLOBAL struct clusinfo cl;
GLOBAL short nsets;
GLOBAL short nclus;
GLOBAL float var[30][12];
int n,index,number,iloop,init;
float clusmean[30][12],totvar[12],dtot[12];
      for(index=0;index <nclus;index++){
              for(init=0;init<12;init++){
                 totvar[init]=0;
                 dtot[init]=0;
                                    }
         for(n=0,number=0;n<nsets;n++){
            if(cl.cluster[n]==index){
              number++;
              for(iloop=0;iloop<12;iloop++){
                 totvar[iloop]=totvar[iloop]+cl.sfeatvars[n][iloop];
                                          }
                                    }
                                     }
              for(iloop=0;iloop<12;iloop++){
                 clusmean[index][iloop]=totvar[iloop]/(float)number;
                                          }
          for(number=0,n=0;n<nsets;n++){
             if(cl.cluster[n]==index){
               number++;
               for(iloop=0;iloop<12;iloop++){
               dtot[iloop]=dtot[iloop]+(cl.sfeatvar[n][iloop]
-clusmean[index][iloop])*(cl.sfeatvar[n][iloop]-clusmean[index][iloop]);
                                          }
                                    }
                                           }
          for(iloop=0;iloop<12;iloop++){
             var[index][iloop]=dtot[iloop]/(float)number;
                                          }
                                  }
}
```

```
Feb 16 13:36:06 1988  varmaxtl.c Page 1
```

Figure AD33

```
#include<std.h>
#include"mylib.h"
varmaxtl() /* To determine which cluster has the largest variance.*/
{
GLOBAL short nclus;
GLOBAL int invarmax,fvarmax;
GLOBAL float var[30][12];
float varmax;
int n,iloop;
for(varmax=0,n=0;n<nclus;n++){
  for(iloop=0;iloop<4;iloop++){/* Only test first four features for clustering*/
      if(var[n][iloop]>varmax){
        invarmax=n;
        fvarmax=iloop;
        varmax=var[n][iloop];
                               }
                                }
                               }
  putfmt("The maximum variance occurs with cluster %i, feature %i .\n",invarmax,fvarmax);
}
```

Feb 16 13:36:06 1988 clusplit.c Page 1

Figure AD34

```
#include<std.h>
#include"mylib.h"
clusplit() /* Increase the number of clusters by splitting the cluster
              with the largest variance or largest distance feature. */
{
GLOBAL struct clusinfo cl;
GLOBAL short nclus,nsets;
GLOBAL int invarmax,fvarmax;
float mnvar,mxvar;
short n,iloop,count;
mnvar=10000;
mxvar=0;
nclus++;
putfmt("Cluster splitting routine entered.\n");
for(n=0;n<nsets;n++){
   if(cl.cluster[n]==invarmax){
     if(cl.featvars[n][fvarmax]>mxvar){
       mxvar=cl.featvars[n][fvarmax];
                                      }
     if(cl.featvars[n][fvarmax]<mnvar){
       mnvar=cl.featvars[n][fvarmax];
                                      }
                                }
                     }
    putfmt("Cluster %i has been split to form cluster %s .\n",invarmax,nclus-1);
/*    putfmt("mxvar= %0.2f, mnvar= %0.2f.\n",mxvar,mnvar); */
     putfmt("Data sets assigned to new cluster are:- \n");
for(ccunt=0,n=0;n<nsets;n++){
   if(cl.cluster[n]==invarmax){
/*    putfmt("cl.featvars[n][fvarmax]-mnvar= (%0.f-%0.f)= %0.f.\n"
,cl.featvars[n][fvarmax],mnvar,cl.featvars[n][fvarmax]-mnvar);
      putfmt("mxvar-cl.featvars[n][fvarmax]= (%0.f-%0.f)= %0.f.\n"
,mxvar,cl.featvars[n][fvarmax],mxvar-cl.featvars[n][fvarmax]); */
     count++;
     if((cl.featvars[n][fvarmax]-mnvar)>(mxvar-cl.featvars[n][fvarmax])){
       putfmt(" %0.f ,",cl.featvars[n][6]);
       cl.cluster[n]= nclus-1;
       if(count % 20 ==0) putfmt("\n");
                                                                        }
     /* else stays with the original index */
                                }
                     }
putfmt("\n");
}
```

Feb 16 13:36:06 1988 clusprint.c Page 1

Figure AD35

```
#include<std.h>
#include"mylib.h"
clusprint() /* Utility routine to print out all existing clusters with */
            /* member cases.                                            */
{
GLOBAL struct clusinfo cl;
GLOBAL short nclus,nsets;
short n,s,tot;
for(n=0;n<nclus;n++){
   putfmt("Members of cluster %s are:-\n",n);
   for(s=0,tot=0;s<nsets;s++){
      if(cl.cluster[s]==n){
        putfmt("%f, ",cl.featvars[s][6]);
        tot++;
        if(tot % 20 ==0){
          putfmt("\n");
                        }
                           }
                        }
   putfmt("\n");
                        }
}
```

Figure AD36

```
#include<std.h>
#include"mylib.h"
move2() /* Move cases to the cluster whose centre is nearest. */
{
GLOBAL struct clusinfo cl;
GLOBAL short nsets,nclus;
static float dist[500][30];
static float tot[12],centot[30][12];
static int ncluster[500];
float dmin;
int i,n,m,s,iloop,move,number;
putfmt("Move routine entered.\n");
for(i=0,move=1;(move ==1)&&(i<30);i++){
   putfmt("Movement loop = %i.\n",i);
   move=0;
   for(n=0;n<nclus;n++){
      for(iloop=0;iloop<12;iloop++){ /* Initialize feature totals. */
         tot[iloop]=0;
                                    }
      for(number=0,s=0;s<nsets;s++){ /* For each set. */
         if(cl.cluster[s]==n){ /* If set is a member of this cluster. */
           number++;
           for(iloop=0;iloop<12;iloop++){
              tot[iloop] += cl.sfeatvars[s][iloop];
/*         putfmt("tot[%i] = %.4f, number = %i.\n",iloop,tot[iloop],number);*/
                                    }
                               }
                                    }
/*       putfmt("For cluster %i coordinates of centre are:- ",n); */
        for(iloop=0;iloop<12;iloop++){
           centot[n][iloop]=(tot[iloop]/(float)number);
/*           putfmt("%.4f, ",centot[n][iloop]); */
                                    }
 /*          putfmt("\n");*/
                                 }
    for(s=0;s<nsets;s++){
       for(n=0;n<nclus;n++){
         dist[s][n]= (centot[n][0]-cl.sfeatvars[s][0])*
                     (centot[n][0]-cl.sfeatvars[s][0])
                    +(centot[n][1]-cl.sfeatvars[s][1])*
                     (centot[n][1]-cl.sfeatvars[s][1])
                    +(centot[n][2]-cl.sfeatvars[s][2])*
                     (centot[n][2]-cl.sfeatvars[s][2])
                    +(centot[n][3]-cl.sfeatvars[s][3])*
                     (centot[n][3]-cl.sfeatvars[s][3]);
                /*    +(centot[n][4]-cl.sfeatvars[s][4])*
                     (centot[n][4]-cl.sfeatvars[s][4])
                    +(centot[n][5]-cl.sfeatvars[s][5])*
                     (centot[n][5]-cl.sfeatvars[s][5]); */
                          }
                       }
for(s=0;(s<nsets)&&(move !=1);s++){
   for(n=0,dmin=1000;n<nclus;n++){ /* Reallocate data sets to nearest cluster. */
      if(dist[s][n]<dmin){
        dmin=dist[s][n];
        ncluster[s]=n;
```



```
                                 }
                                    }
         if(ncluster[s] != cl.cluster[s]){
           move=1;
           putfmt("Cluster change for data set %f from cluster %i to %i.\n"
,cl.featvars[s][6],cl.cluster[s],ncluster[s]);
           cl.cluster[s]=ncluster[s];
                                    }
/*    putfmt("For data set %i nearest cluster is:- %i.\n",s,ncluster[s]); */
                                    }
                                 }
}
```

Figure AD37


```
#include<std.h>
#include"mylib.h"
crit2() /* Function to find the cluster with the largest pick-up point */
        /* range and test this against a maximum threshold.           */
{
GLOBAL struct clusinfo cl;
GLOBAL short nsets,nclus,mnrange,crange;
GLOBAL int invarmax;
GLOBAL short maxDtoM[3][30];
GLOBAL short minDtoM[3][30];
short range[3][30];
short nc,ns,t,index,iloop,r,maxrange,cindex;
if(nclus>=29){
  return(-1);
            }
maxrange=0;
for(t=0;t<30;t++){
   for(n=0;n<3;n++){
      maxDtoM[n][t]=0;
      minDtoM[n][t]=1000;
      range[n][t]=0;
                  }
               }
/* First find the mins and maxs of distances to centre in each cluster. */
for(nc=0;nc<nclus;nc++){
   for(ns=0;ns<nsets;ns++){
      if(cl.cluster[ns]==nc){
        for(index=0,iloop=9;iloop<12;iloop++,index++){
           if(cl.featvars[ns][iloop]>maxDtoM[index][nc]){
             maxDtoM[index][nc]=cl.featvars[ns][iloop];
                                                      }
           if(cl.featvars[ns][iloop]<minDtoM[index][nc]){
             minDtoM[index][nc]=cl.featvars[ns][iloop];
                                                      }
                                                    }
                                 }
                              }
                           }
for(nc=0;nc<nclus;nc++){
   for(index=0;index<3;index++){
      range[index][nc]=maxDtoM[index][nc]-minDtoM[index][nc];
      putfmt("For cluster %s,feature %s, range is %s.\n",nc,index,range[index][nc]);
      if(range[index][nc]>maxrange){
        maxrange=range[index][nc];
        cindex=index;
        crange=nc; /* The index of the cluster to be split.*/
                                   }
                                }
                           }
  putfmt("The maximum range (%s) occurs for cluster %s, feature %s, range=%s.\n"
,mnrange,crange,cindex,maxrange);
if(maxrange<=mnrange){
  return(-1);
                     }
else {
      invarmax=crange;
```



```
      return(0);
     }
}
```

Figure AD38

```
Feb 16 13:36:06 1988  varmaxt2.c Page 1

#include<std.h>
#include"mylib.h"
varmaxt2() /* To determine the variance and feature with this largest    */
           /* variance in the cluster with the largest distance feature. */
{
GLOBAL struct clusinfo cl;
GLOBAL short nclus,nsets;
GLOBAL int invarmax,fvarmax;
GLOBAL float var[30][12];
float minf[4],maxf[4],range[4];
float varmax,totvar,mr;
int n,iloop,ns;
n=invarmax;
varmax=0;
totvar=0;
mr=0;
for(iloop=0;iloop<4;iloop++){
   minf[iloop]=1000;
   maxf[iloop]=0;
   range[iloop]=0;
   totvar += var[n][iloop];
                             }
if(totvar==0){
  putfmt("The total variance of all features in this cluster is zero.\n");
  for(ns=0;ns<nsets;ns++){
     if(cl.cluster[ns]==invarmax){
       for(iloop=0;iloop<4;iloop++){
          if(cl.sfeatvars[ns][iloop]>maxf[iloop]){
            maxf[iloop]=cl.sfeatvars[ns][iloop];
                                                  }
          if(cl.sfeatvars[ns][iloop]<minf[iloop]){
            minf[iloop]=cl.sfeatvars[ns][iloop];
                                                  }
                                      }
                                    }
                                  }
  for(iloop=0;iloop<4;iloop++){
     range[iloop]=(maxf[iloop]-minf[iloop]);
     if(range[iloop]>mr){
       fvarmax=iloop;
       varmax=0;
                         }
                                 }
            }
else {
   for(iloop=0;iloop<4;iloop++){
      if(var[n][iloop]>varmax){
        fvarmax=iloop;
        varmax=var[n][iloop];
                               }
      }
      putfmt("The maximum variance of cluster %i, is %2.4f and occurs with feature %i.\n"
,invarmax,varmax,fvarmax);
}
```

Figure AD39

```
#include<std.h>
#include"mylib.h"
cetinfo() /* Routire to determine the cluster information to be stored. */
{
GLOBAL struct clusinfo c1;
GLOBAL short nclus,nsets;
GLOBAL short maxvar[30][12],minvar[30][12];
GLOBAL short npoints[30],ptlist[30][400];
GLOBAL short maxDtoM[3][30];
GLOBAL short minDtoM[3][30];
GLOBAL short cenD[3][30];
int ci,n,f,a,h,iloop;
short np;
/* Build up an array of maxes and mins for each feature. */
for(a=0;a<30;a++){
   for(b=0;b<12;b++){
      minvar[a][b]=10000;
      maxvar[a][b]=0;
                    }
                  }
for(ci=0;ci<nclus;ci++){ /* for each cluster */
   for(iloop=0;iloop<3;iloop++){ /* for each distance feature.*/
      cenD[iloop][ci]=(maxDtoM[iloop][ci]+minDtoM[iloop][ci])/2;
putfmt("maxD= %s, minD= %s, cenD= %s.\n",maxDtoM[iloop][ci],minDtoM[iloop][ci],cenD[iloop][ci]);
      /* This takes av. dist maybe better with median or mode. */
                              }
   for(n=0;n<nsets;n++){ /* for each set of data */
      if(c1.cluster[n]==ci){ /* if data set is a member of cluster */
        np=npoints[ci]++; /* increment cluster point count */
        ptlist[ci][np]=c1.featvars[n][6]; /* append point to list of points */
        for(iloop=0;iloop<9;iloop++){
           if(c1.featvars[n][iloop]>maxvar[ci][iloop]){
             maxvar[ci][iloop]=c1.featvars[n][iloop];
                                                      }
           if(c1.featvars[n][iloop]<minvar[ci][iloop]){
             minvar[ci][iloop]=c1.featvars[n][iloop];
                                                      }
                                    }
                           }
                          }
putfmt("For cluster %s number of points is: %s.\n",ci,np);
putfmt("Feature max and mins are:-\n");
for(iloop=0;iloop<9;iloop++){
   putfmt("%s, %s.  ",maxvar[ci][iloop],minvar[ci][iloop]);
                            }
putfmt("\n The distances to the centre are:- %s,%s,%s.\n",cenD[0][ci],cenD[1][ci],cenD[2][ci]);
                  }
}
```

Figure AD40

```
#include<std.h>
#include"mylib.h"
storfile() /* Store on disc in a specified file on the 68020 the cluster and */
           /* orientation information necssary tc generate the cluster map.  */
{
extern struct FILFCATA fdata,*fss;
GLCBAL struct clusinfo cl;
GLCBAL short nclus,nsets,mnclus;
GLCBAL short maxvar[30][12],minvar[30][12];
GLCBAL short npoints[30],ptlist[30][400];
GLCBAL short maxDtoM[3][30],minDtoM[3][30];
GLCBAL short cenD[3][30];
short buf[1000];
short n,ci,ns,iloop,rp,q;
int drive;
char fti,ru,*pframe,frame[10],iname[10];
drive=1;
q=1;
pframe=fname;
putfmt("Enter the rame of the cluster data storage file.\n");
getfmt("%n",fname);
putfmt("The data is to be stored in file \"%p\".\n",fname);
putfmt("Enter a data description for the file.\n");
getfmt("%p",iname);
putfmt("The file is described as \"%p\".\n",iname);
putfmt("Enter a file type index.\n\n");
putfmt("               1 for a bowl file.\n");
putfmt("               2 for a plate file.\n");
putfmt("               3 for a cup file.\n");
putfmt("               4 for another item type.\n\n");
getfmt("%c",&fti);
putfmt("How many 0.9 degree units were made per rotation step.\n");
getfmt("%c",&ru);
putfmt("There are %s 0.9 degree rotation units per step.\n",ru);
/* outfmt("At point 1.\n"); */
buf[q++]=fti;
/* outfmt("At point 2.\n"); */
for(n=0;n<10;n++){
   buf[q++]=iname[n];
                 }
buf[q++]=ru;
buf[q++]=nclus;
for(ci=0;ci<nclus;ci++){
   buf[q++]=ci;
   for(n=0;n<9;n++){
      buf[q++]=minvar[ci][n];
      buf[q++]=maxvar[ci][n];
                    }
   for(n=0;n<3;n++){
      buf[q++]=cenD[n][ci];
                    }
   buf[q++]=npoints[ci];
   for(np=0;np<npoints[ci];np++){
      buf[q++]=ptlist[ci][np];
                                }
                       }

/* Number of 512 byte sets of data to be o/p is c/256+1 */
buf[0]=q-1; /* Total number of sets of cata to be o/p */
fdata.pbuf=buf;
fdata.count=q/256+1;
/* outfmt("Number of buffer elements= %s, q/256= %s.\n",q,q/256+1); */
for(n=0;n<15;n++){
   fdata.name[n]=fname[n];
                 }
if((n=system(7,&drive))!=0){
  putfmt("Error in accessing drive 1.\n");
                           }
if((n=system(16))!=0){
  putfmt("Error in restoring drive 1.\n");
                     }
if((n=system(9,fss))!=0){
  putfmt("Error in opening the file.\n");
  putfmt("Disc storage sequence aborts.\n");
  goto end;
                        }
if((n=system(14,fss))!=0){ /* Write out q/256 blocks of cata */
  putfmt("Error in writing to disc file %p.\n",fdata.name);
                         }
if((n=system(11,fss))!=0){
  putfmt("Error in closing file.\n");
                         }
end:
system(9);
putfmt("Drive 1 deselected.\n");
putfmt("Storage sequence completed.\n");
}
```

Figure AD41

```
#include <std.h>                    /* include standard Whitesmith's header */
/* declaration and initialization of external variables */
short *ppoint=pcint;                /* line element pointer  */
char *PMPIXEL=mpixel;               /* image element pointer */
short *line[100]=pcint;             /* array of pointers to line starts */
short *endline[100]=pcint;          /*                      line ends   */
short *aline[100]=point;            /* alternative          line starts */
short *aendline[100]=point;         /* alternative          line ends   */
char mpixel[53761]=0;               /* exparded image array             */
short cimpoint[100][25]=0;          /* line number indexed feature array*/
short impoint[100][10]=0;           /*             imporant line points */
/* short timpoint[300][15]=0;*/ /*                   output features*/
short spoint[300]=0;
char screen[7681]=0;                /* small received image array       */
short point[30000]=0;               /* secuential line storage array    */
short bound[100]=0;                 /* line indexed boundary point array*/
short totarea[100]=0;               /*              area          array */
short totperi[100]=0;               /*              perimeter           */
short paction[100]=0;               /*              compaction          */
unsigned short type[100][3]=0;      /*              shape type          */
short outline[100][30]=0;           /*                                  */
short minpid=0;                     /*   important points               */
short midpid=0;
short maxpid=0;
short LINE=0;                       /* line index                       */
short FI=0;                         /* frame index                      */
short ascale=0;                     /* lateral scaling factor for area  */
short J=0;                          /*   sundry variables               */
short I=0;
int pindex=0;
int LS=0;
short N=0;
short BP=0;
short fd=0;
short impl=0;
short lilen=50;                     /* optional minimum line length considered */

/* External variables used in DOS image and data file accesses          */
struct FILEDATA                     /* DOS file structure support       */
        {
        char name[16];
        int block;
        int load;
        int start;
        int count;
        char *pbuf;
        int fsize;
        };
struct FILEDATA fdata1= {
        "++++++++++++++++",
        0,0,0,1,NULL,0
                        };
struct FILEDATA fdata2= {
        "++++++++++++++++",
        0,0,0,1,NULL,0
                        };
```



```
struct FILEDATA *fss1=NULL;
struct FILEDATA *fss2=NULL;
short databuf[1000]=0;              /* data buffer for reading in file data  */
int ni=1000;                        /* number of images                      */
int namin=0;                        /* index to file name                    */
int mult=0;                         /* interval in file index                */
int RI=1;                           /* main loop index                       */
```

Figure AD41 Continued.

```
/* External variables used in cluster map generation                    */
unsigned short MAP[0x40000]=0;/* actual cluster look-up table           */
unsigned char col[256][10]=0; /* collision record table                 */
float div[4]=0;               /* scaling ratio for features             */
short minfe[4]=0;             /* min of features from each data file    */
short maxfe[4]=0;             /* max                                    */
short fetol[4]=0;             /* tolerance in feature mapping stage     */
short maxv1[8][20][10]=0;     /* cluster mins and maxs                  */
short minv1[8][20][10]=0;
short npoints[8][20]=0;       /* number of image orientations/cluster   */
short ptlist[8][20][500]=0;   /* list of angular images/cluster         */
short nclus[8]=0;             /* number of clusters per file            */
short cend[8][20][3]=0;       /* distances to centres of ccluster       */
short clusi[8][20]=0;         /*                                        */
char ftype[8]=0;              /* type coding of file                    */
char ru[8]=0;                 /* rotation units of 0.9 deg between imag*/
char fname[8][16]=0;          /* item data file names                   */
char iname[8][10]=0;          /* item data file internal description    */
short nfil=0;                 /* number of item data files considered   */
/*
short mincir[8][20][200][2]=0;  look-up of points on centre radii
short midcir[8][20][200][2]=0;
short maxcir[8][20][200][2]=0;
*/
short nmincir[200][2]=0;      /* absolute circumference coordinates     */
short nmidcir[200][2]=0;
short nmaxcir[200][2]=0;
short indmin[8][20]=0;        /* index to points on center circle peri*/
short indmid[8][20]=0;
short indmax[8][20]=0;
short cross[10][4][2][14]=0;  /* corners of centre crosses to be o/p    */
short crossind=0;             /* index for circle intercepts            */
unsigned char afile[100][5]=0; /* data file id associated with lines    */
unsigned char acluster[100][5]=0; /* cluster id for identified lines    */

/* External variables used in setting program options */
short MAPTEST=0;              /* is a clustermap test required          */
short SCLUS=0;                /* screen pres of clusters in data file   */
short IMSOURCE=0;             /* test image source option               */
short PRINTOR=0;              /* print original image option            */
short PRINTHIN=0;             /* print thinned image option             */
short PRINTFT=0;              /* print feature table option             */
short PLOTCORN=0;             /* plot boundary coded line on DUET       */
short PLOTCEN=0;              /* plot "cocked hat" corners on DUET      */
short PLOTCIR=0;              /* plot intercept circles on DUET         */
short HCRES=0;                /* print hardcopy of identification res   */
short EOS=0;                  /* end of plot secuence marker for DUET   */
short PRINTSET=0;             /* print destination marker 0=console     */
```



```
short WL=0;                   /* wait loop variable for DUET o/p        */
short CLDATA=0;               /* cluster data file information print    */
short PSPEED=0;               /* option setting flag for no-print       */
short SPEED=0;                /* maximum speed no-print o/p flag        */
main()          /* coordinating function for full scene analysis from disc      */
                /* image storage on 68020 system with feature transfer to duet. */
{
GLOBAL short EOS,IMSOURCE,PRINTOR,PRINTHIN,PRINTFT,PLOTCORN,PLOTCEN,HCRES;
GLOBAL short PLOTCIR,PRINTSET,SPEED,PSPEED;
GLOBAL short minpid,midpid,maxpid,J,I,N,BP,fd,lilen,*ppoint;
GLOBAL int pindex,RI,ni;
GLOBAL short bound[100];
GLOBAL short totperi[100];
GLOBAL short totarea[100];
GLOBAL short paction[100];
GLOBAL short cimpoint[100][25];
GLOBAL short impoint[100][10];
GLOBAL unsigned short type[100][3];
GLOBAL short spoint[300];
GLOBAL short outline[100][30];
GLOBAL short point[30000];
int aquipwd();
register short c,n;
register short *pgen1,*pgen2;
short s,n;
char getch();
int selshad();
PRINTSET=0;
```

```
cache();                                /* set cache running on 68C20 board    */
initdu();                               /* initialize 68C20 on-board UART      */
greet();
changep();
putfmt("Program operating parameters and input/output options set.\n\n");
lookup();                           /* generate object i.d. look-up table  */
PRINTSET=0;
putfmt("Object class look-up table generated.\n\n");
if(IMSOURCE==1){
  filepar();                            /* read in the file processing details */
               }
else if(IMSOURCE==2){
       apar();
                    }
for(RI=1;RI<=ni;RI++){
   if(IMSOURCE==1){
     if((RI==31)||(RI==61)||(RI==91)||(RI==121)||(RI==151)||(RI==181)){
       PRINTSET=0;
       putfmt("Change the disc if necessary.\n\n");
       getch();getch();
                                                                       }
                    }
for(c=0;c<100;c++){                     /* reinitialization of global variables */
   bound[c]=0;
   totperi[c]=0;
   totarea[c]=0;
   paction[c]=0;
   for(m=0;m<3;m++){
```

Figure AD41 Continued.


```
      type[c][m]=0;
                   }
   for(m=0;m<25;m++){
      cimocint[c][m]=0;
                    }
   for(m=0;m<10;m++){
      impoint[c][m]=0;
                    }
   for(m=0;m<30;m++){
      type[c][m]=0;
      outline[c][m]=0;
                    }
                  }
for(pgen1=point,m=0;m<6000;m++){
   *(pgen1++)=0;
   *(pgen1++)=0;
   *(pgen1++)=0;
   *(pgen1++)=0;
   *(pgen1++)=0;
                               }
for(pgen1=socint,m=0;m<100;m++){
   *(pgen1++)=0;
   *(pgen1++)=0;
   *(pgen1++)=0;
                  }
aline[0]=point;
endline[0]=point;
aendline[0]=point;
pocint=point;
PMPIXEL=mpixel;
line[0]=point;
LINE=0;
minpid=0;
micpid=0;
maxpid=0;
J=0;
I=0;
circex=0;
N=0;
BP=0;
fd=0;
PRINTSET=0;
    if(IMSOURCE==1){
      if(aquipwd()==(-1)){ /* Test for image read error */
        goto terminate;
                         }
      putfmt("Disc image received.\n\n");
                        }
    else if(IMSOURCE==2){
           putfmt("Awaiting image input from the Apple.\n");
           aquipwap();
           putfmt("Image received from Apple.\r\n");
                        }
    if(PSPEED==1)SPEED=1;
```

```
    bitmatpw();
    putfmt("Expansion of bits into bytes completed.\n\n");
```

Figure AD41 Continued.


```
    if(PRINTOR==1){
      pripw();
                  }
kon();
putfmt("Edge extraction completed.\r\n");
if(PRINTHIN==1){
  pripw();
                }
for(N=0;(((s=selshad()) !=0)&&(N !=100));N++){
   outfmt(".");
                                              }
 putfmt("\n");
 putfmt("Chain coding sequence completec.\r\n");
 fclosed();
 putfmt("Test for closed outlines completeo.\n\r");
 xend();
 putfmt("Artificial end of boundary generation completed.\n\n");
 shadw();
 putfmt("Open line feature point derivation completeo.\n\n");
 cimp();
 putfmt("Open line feature calculation completeo.\n\n");
 fperi();
 putfmt("Closec boundary perimeter calculation completed.\n");
 farea();
 outfmt("Closeo boundary area calculation completed.\n");
 arcorrect();
 putfmt("Lateral area correction completed.\n");
 compact();
 putfmt("Closeo boundary compaction-factor calculation completed.\n\n");
 if(HCRES==1){
   SPEED=0;
   PRINTSET=1;
             }
 identify();
if(PSPEED==1)SPEED=1;
 PRINTSET=0;
 putfmt("Object identification stage completed.\n\n");
 circept();
 putfmt("Intersection point of endpoint radii calculated.\n\n");
 cockhat();
 putfmt("Centre of gravity of intersection radii calculated.\n\n");
 if(PRINTFT==1){
   printable();
                }
 if(PLOTCORN==1){
   shipcornpw();
                 }
/*    if(PLOTCIR==1){
   circle();
   putfmt("Intercept circle drawing routine finished.\n");
                } */
 if(PLOTCEN==1){
   shipcen();
               }
SPEED=0;
       }
```



```
putfmt("\nExiting from program after processing specified images.\n");
exit();
terminate:
putfmt("\n Program terminated due to disk read error.\n");
exit();
}
```

```
#include<std.h>
#include"mylib.h"
lookup()        /* 68020 main testing program look-up table generator */
{
GLOBAL short MAPTEST,SCLUS,PRINTSET,PLOTCIR;
putfmt("\n\n");
putfmt("       ************************************************\n");
putfmt("       *                                              *\n");
putfmt("       *          LOCK-UP TABLE GENERATION SECTION    *\n");
putfmt("       *                                              *\n");
putfmt("       ************************************************\n");
putfmt("\n\n\n");
PRINTSET=0;
rdbkdata();     /* Read in specified item cluster data files          */
putfmt("Specifiec cluster data files reac in.\r\n");
PRINTSET=0;
/*
if(PLOTCIR==1){
  circalcs();    Calculate the relative positions of points on
                 circle centre perimeters.
  putfmt("Encpoint relative pick-up point arc look-up calculations made.\n\n");
               }
*/
if(SCLUS==1){
clscreen();
putfmt("Cluster data file information displayed on screen.\n\n");
            }
PRINTSET=C;
map();  /* Create the cluster map for giver item data.        */
putfmt("Cluster data look-up table created.\n\r");
if(MAPTEST==1){
PRINTSET=0;
maptest();      /* Function to test data map procucec.                */
putfmt("Cluster map testing complete.\n\r");
              }
}
```

Figure AD42


```
#include<std.h>
#include"mylib.h"
rdbkdata() /* To readback cluster data file on 68C2C system. */
{
GLOBAL struct FILEDATA fdata1,*fss1;
GLOBAL short PRINTSET,CLDATA;
GLOBAL short maxv1[9][20][10],minv1[8][20][10];
GLOBAL short npoints[9][20],plist[8][20][500];
GLOBAL short nfil,rclus[8];
GLOBAL short cenC[8][2C][2];
GLOBAL short clusi[8][2C];
GLOBAL char frame[8][16],iframe[8][10];
GLOBAL char ftype[8],ru[8];
GLOBAL short databuf[1CCC];
short iloop,r,t,f,ci,o,rcb;
int ind;
int pcount;
int system();
int drive,nr;
int i1,i2,i3,i4;
char *pframe;
fss1= &fdata1;
crive=1;
putfmt("Cluster data file inputting section.\n\n");
putfmt("Enter the number of cluster data files to be processed.\n");
getfmt("%s",&nfil);
for(f=C;f<nfil;f++){
   outfmt("Enter the name of the next cluster file to be processed\n");
   putfmt("corresponding to an item to be identifiec.\n");
   pframe=(&frame[f][0]);
   getfmt("%p",pframe);
                    }
for(f=0;f<nfil;f++){
   o=1;
   putfmt("File number %s to be read in is \"%p\".\n",f,&frame[f][0]);
   if((n=system(7,&drive))!=0){
     outfmt("Error in accessing drive 1.\n");
                              }
   if((n=system(16))!=C){
     outfmt("Error in restoring drive 1.\n");
                         }
   for(n=C;n<15;n++){
      fdata1.name[r]=frame[f][n];
                    }
   if((n=system(1C,fss1))!=0){
     outfmt("Error in opening file %p for read.\n",&fdata1.name[0]);
     goto end;
                             }
   fdata1.pbuf=(char *)databuf;
   fdata1.count=1;
   if((n=system(13,fss1))!=0){
     outfmt("Error in reading file.\n");
     goto end;
                             }
pccunt=0;
  rob=databuf[0]+2;
```

Figure AD43

Figure AD43 Continued.

```
/* i1=fdata1.ccunt;
i2=((int)(fdata1.pbuf)-(int)&(databuf[0]));
i3=fss1;
i4=(int)(fdata1.pbuf);
putfmt("nob=%s, count=%i, fdata.pbuf=%hi, pbuf-buf =%i, fss1=%hi.\n",nob,i1,i4,i2,i3); */
  if(nob<=510)fdata1.ccunt=0;
  else {
       (fdata1.ccurt= (nob+2)/512);
       fdata1.pbuf +=512;
       if((n=system(13,fss1))!=0){
         putfmt("Error in reading file.\r");
         goto end;
                                  }
       }
if(CLDATA==1){
  PRINTSET=1;
  putfmt("File number %s to be read in is \"%p\".\n",f,&frame[f][0]);
             }
else PRINTSET=0;
 /* Allocate data in the databuffer to its respective respective variables */
while(q <=databuf[0]){
     ftype[f]=(char)databuf[q++];
       if(ftype[f]==1){
         putfmt("This file is a bowl file.\n");
                      }
       else if(ftype[f]==2){
            putfmt("This file is a plate file.\n");
                           }
       else if(ftype[f]==3){
            putfmt("This file is a cup file.\n");
                           }
       else {
            putfmt("File number %s is arcther item ftype.\r",f);
            }
     for(n=0;n<10;r++){
        frame[f][n]=databuf[q++];
                       }
     putfmt("The description of the file being processed is \"%p\" file.\n",&frame[f][0]);
     ru[f]=(char)databuf[q++];
     nclus[f]=databuf[q++];
     putfmt("The number of clusters in this file is %s.\r",nclus[f]);
     for(ci=0;ci<nclus[f];ci++){
        clusi[f][ci]=databuf[q++];
          putfmt("Cluster index is: %s.\n",clusi[f][ci]);
          for(iloop=0;iloop<9;iloop++){
             minvl[f][ci][iloop]=databuf[q++];
             maxvl[f][ci][iloop]=databuf[q++];
putfmt("For feature %s, mirimum = %s, maximum = %s.\n",iloop,minvl[f][ci][iloop],mexvl[f][ci][ilccc]);
                                        }
            for(iloop=0;iloop<3;iloop++){
              cen0[f][ci][iloop]=databuf[q++];
              putfmt("Centre distances are: %s.\n",cen0[f][ci][iloop]);
                                         }
       npoints[f][ci]=databuf[q++];
       putfmt("The number of images in this cluster is %s.\n",npoints[f][ci]);
       putfmt("The image incices are: ");
```



```
       for(n=0;n<npoints[f][ci];n++){  /* for each point */
            ptlist[f][ci][n]=databuf[q++];
            putfmt("%s.",ptlist[f][ci][n]);
            if((n % 12)==0)putfmt("\n");
                                       }
        putfmt("\n\n");
                    }
                  }
     }
system(8);
putfmt("Drive 1 deselected.\n");
PRINTSET=0;
return(0);
enc:
putfmt("Cluster data file accessing error.\n");
system(8);
putfmt("Drive 1 deselected.\n");
PRINTSET=0;
return(-1);
}
```

Figure AD44

```
#include<std.h>
clscreen()
{
GLOBAL short maxvl[8][20][10];
GLOBAL short minvl[8][20][10];
GLOBAL short cenD[8][20][3];
GLOBAL short nfil,nclus[8];
GLOBAL short PRINTSET;
short maxfe[4],minfe[4],felow[4],sfelow[4],fehi[4],sfehi[4];
float range[4],div[4];
char cluscode;
short fei,ci,fil,col,df;
PRINTSET=1;
putfmt("\n\n\n");
putfmt("Printout of Cluster Data Information Read-in From Disk.\n\n");
for(fei=0;fei<4;fei++){
   maxfe[fei]=0;
   minfe[fei]=10000;
                      }
for(fil=0;fil<nfil;fil++){
   putfmt("For file %s.\n",fil);
   for(ci=0;ci<nclus[fil];ci++){
      putfmt("  For cluster %s.\n",ci);
      for(fei=0;fei<4;fei++){
         if(maxvl[fil][ci][fei]>maxfe[fei]){
           maxfe[fei]=maxvl[fil][ci][fei];
                                           }
         if(minvl[fil][ci][fei]<minfe[fei]){
           minfe[fei]=minvl[fil][ci][fei];
                                           }
                            }
        putfmt("Ranges of features are: compaction %s,%s, minmax %s,%s, minmid %s,%s, midmax %s,%s.\n"
,minvl[fil][ci][0],maxvl[fil][ci][0],minvl[fil][ci][1],maxvl[fil][ci][1],minvl[fil][ci][2],
maxvl[fil][ci][2],minvl[fil][ci][3],maxvl[fil][ci][3]);
                               }
                          }
for(fei=0;fei<4;fei++){
   range[fei]=(float)(maxfe[fei]-minfe[fei]);
   div[fei]=range[fei]/64;
                       }
putfmt("Ranges of identification features are: compaction= %s,%s, minmax= %s,%s, minmic=%s,%s, midmax=%s,%s.\n"
,minfe[0],maxfe[0],minfe[1],maxfe[1],minfe[2],maxfe[2],minfe[3],maxfe[3]);
for(fil=0;fil<nfil;fil++){
   putfmt("For file number %s.\n",fil);
   for(ci=0;ci<nclus[fil];ci++){
        putfmt("For cluster %s, the distances of the endpoints to the pick-up point are: %s, %s, %s.\n"
,ci,cenD[fil][ci][0],cenD[fil][ci][1],cenD[fil][ci][2]);
                               }
                          }
putfmt("Printout Representation of Feature Ranges.\n\n");
for(fei=0;fei<4;fei++){
   putfmt("\n");
   if(fei==0){
     putfmt("COMPACTION.\n\n");
             }
   if(fei==1){
     putfmt("MINMAX.\n\n");
             }
   if(fei==2){
     putfmt("MINMID.\n\n");
             }
   if(fei==3){
     putfmt("MIDMAX.\n\n");
             }
   for(fil=0;fil<nfil;fil++){
      for(ci=0;ci<rclus[fil];ci++){
         putfmt("Cluster number is %s.\n",ci);
         cluscode=(char)(32*fil+ci);
         felow[fei]=minvl[fil][ci][fei]-minfe[fei];
         sfelow[fei]=(short)((float)felow[fei]/div[fei]);
         fehi[fei]=maxvl[fil][ci][fei]-minfe[fei];
         sfehi[fei]=(short)((float)fehi[fei]/div[fei]);
    putfmt("felow=%s, sfelow=%s, fehi=%s, sfehi=%s.\n"
,felow[fei],sfelow[fei],fehi[fei],sfehi[fei]);
         for(col=0;col<=64;col +=2){
            if((sfelow[fei]<=col)&&(sfehi[fei]>=col)){
              putfmt("%2hc",cluscode);
                                                     }
            else if((sfelow[fei]<=col+1)&&(sfehi[fei]>=col+1)){
              putfmt("%2hc",cluscode);
                                                               }
            else putfmt("++");
                                }
         putfmt("\n\n");
                              }
                         }
                    }
}
```

Figure AD45

```
#include<std.h>
map() /* To build up cluster map from item file data (collision version).*/
{
GLOBAL unsigned short MAP[0x40000]; /* Try with a 256*2 K byte array.    */
GLOBAL unsigned char col[256][10];  /* collision array                   */
GLOBAL short maxv1[8][20][10];      /* maxs of features (file/clus/var)  */
GLOBAL short minv1[8][20][10];      /* mins                              */
GLOBAL short nfil,nclus[8];         /* no. of files, number of clus/fil  */
GLOBAL float div[4];                /* feature scaling factor            */
GLOBAL short maxfe[4],minfe[4],fetol[4]; /* overall feature maxs & mins  */
GLOBAL short PRINTSET;
short felow[4],fehi[4];
float range[4];
short cmp,mnx,mnmd,mdmx,addcmp,addmnx,addmnmd;
short baddcmp,baddmnx,baddmnmd,selement,celement;
int baddmdmx;
int addmdmx;
short fil,ci,fei,n,m,t,tt,found,index,count,tadd,rtad;
unsigned int add,totadd,btotadd,colindex;
unsigned short indval;
char cluscode,code,number;
char sequence[10];
unsigned char collid=0xff;
if(PRINTSET==1){ /* set printer to elite size */
  putfmt("\033M");
               }
colindex=0;
baddcmp=baddmnx=baddmnmd=baddmdmx=-1;
btotadd=0xffff;
putfmt("Cluster-mapping routine entered.\n\n");
for(fei=0;fei<4;fei++){
   maxfe[fei]=0;
   minfe[fei]=10000;
                      }
/* Get maxes and mins of each feature */
for(fil=0;fil<nfil;fil++){                /* For each file */
   for(ci=0;ci<nclus[fil];ci++){          /* For each cluster */
      for(fei=0;fei<4;fei++){            /* For each feature */
         if(maxv1[fil][ci][fei]>maxfe[fei]){
           maxfe[fei]=maxv1[fil][ci][fei];
                                           }
         if(minv1[fil][ci][fei]<minfe[fei]){
           minfe[fei]=minv1[fil][ci][fei];
                                           }
                           }
                              }
                          }
putfmt("For all input data files and clusters the features minima and maxima are:-\n");
for(fei=0;fei<4;fei++){ /* Print max and mins for each feature */
   putfmt("For feature %s minimum= %s, maximum= %s.\n",fei,minfe[fei],maxfe[fei]);
                      }
for(fei=0;fei<2;fei++){ /* 32 point range for compaction and minmax */
   range[fei]=(float)((maxfe[fei]+fetol[fei])-(minfe[fei]-fetol[fei]));
   div[fei]=range[fei]/31; /* N-1 for feature to give correct range */
   putfmt("For feature %s, fetol=%s, range= %0.2f, div= %0.2f.\n",fei,fetol[fei],range[fei],div[fei]);
                      }
for(fei=2;fei<4;fei++){ /* 16 point range for minmid and midmax     */
   range[fei]=(float)((maxfe[fei]+fetol[fei])-(minfe[fei]-fetol[fei]));
   div[fei]=range[fei]/15; /* N-1 for feature to give correct range */
   outfmt("For feature %s, fetol=%s, range= %0.2f, div= %0.2f.\n",fei,fetol[fei],range[fei],div[fei]);
                      }
/* putfmt("Divisors for features are:-%0.2f, %0.2f, %0.2f, %0.2f.\n",div[0],div[1],div[2],div[3]); */
/* Now we have the offset and division factor */
/* Now build up map by writing cluster codes to each mapped location.*/
for(fil=0;fil<nfil;fil++){                          /* For each file */
   for(ci=0;ci<nclus[fil];ci++){                    /* For each cluster */
      putfmt("New cluster ci=%s.\n",ci);
      for(fei=0;fei<4;fei++){                       /* For each feature */
      felow[fei]=(minv1[fil][ci][fei]-minfe[fei])-fetol[fei];
      fehi[fei]=(maxv1[fil][ci][fei]-minfe[fei])+fetol[fei];
                            }
   for(baddcmp= -1,cmp=felow[0];cmp<=fehi[0];cmp++){ /* Indexing compaction */
      /* putfmt("New compaction loop, cmp= %s.\n",cmp); */
      addcmp=(short)((float)cmp/div[0]):            /* Scaled compaction  */
      if(addcmp==baddcmp)continue; /* Skip remainder of loop */
      baddcmp=addcmp;
      for(baddmnx= -1,mnx=felow[1];mnx<=fehi[1];mnx++){ /* Indexing minmax */
         addmnx=((short)((float)mnx/div[1]))<<5; /* Multiply by 32 */
         if(addmnx==baddmnx)continue;
         baddmnx=addmnx;
         for(baddmnmd= -1,mnmd=felow[2];mnmd<=fehi[2];mnmd++){ /* Indexing
                                                                   minmid */
            addmnmd=((short)((float)mnmd/div[2]))<<10; /* Multiply by 256*4 */
            if(addmnmd==baddmnmd)continue;
            baddmnmd=addmnmd;
            for(baddmdmx= -1,mdmx=felow[3];mdmx<=fehi[3];mdmx++){
                                                   /* Indexing midmax */
          /* putfmt("mdmx= %s,felow= %s, fehi= %s.\n",mdmx,felow[3],fehi[3]); */
              addmdmx=((int)((float)mdmx/div[3]))<<14; /* Mltply by 64*256 */
              if(addmdmx==baddmdmx)continue;
              baddmdmx=addcmdmx;
              add=(addcmp+addmnx+addmnmd+addmdmx);
              totadd=add;
              totadd=(totadd & 0x3ffff);
              cluscode=(char)(32*(fil+1)+ci); /* 3 MSB file, 5 LSB cluster. */
```

```
/* putfmt("addcmp= %hs, addmnx= %hs, addmnmd= %hs, addmdmx= %hi, totadd= %hi.\n"
,addcmp,addmnx,addmnmd,addmdmx,totadd); */
                /* putfmt("%hi. ",totadd); */
                celement=(*(MAP+totadd)); /* Element addressed */
                selement=(*(MAP+totadd));
/* putfmt("cluscode= %hc, celement= %hs.\n",cluscode,celement); */
                if((celement & collid)==0){
                  *(MAP+totadd)=cluscode;
                  if(cluscode !=code){
                    putfmt("code= %hc.\n",cluscode);
                                      }
                  code=cluscode;
                  /* putfmt("MAP location set to cluscode(%hc).\n",cluscode); */
                        }
                else if((celement & collid)!=collid){ /* A first time collision. */
/* putfmt("cluscode= %hc, celement= %hs.\n",cluscode,celement); */
/*                      putfmt("A first time collision.\n"); */
                     for(n=0,found=0;(n<colindex)&&(found !=1);n++){ /* Does this collision */
                                                                    /* already exist.     */
                       if((col[n][0]==celement)&&(col[n][1]==cluscode)){
                         found=1;
                         indval=(256*n)+collid;
                         *(MAP+totadd)=indval;
/*       putfmt("Cluster already exists. n= %s. (256*n)=%hs.\n",n,(256*n)); */
                                                                        }
                                                                    }
                     /* If no cluster found create a new one */
                     if(found==0){
          putfmt("No cluster found, new one created, colindex= %i.\n",colindex);
                       col[colindex][0]=celement;
                       col[colindex][1]=cluscode;
                       indval=(256*colindex)+collid;
                       *(MAP+totadd)=indval;
              /*         putfmt("Value written to map is %hs. (256*n)=%hs.\n",indval,(256*r)); */
                       colindex++;
                                  }
                 else if(celement==collid){ /* A collision already occured here */
/* putfmt("cluscode= %hc, celement= %hs.\n",cluscode,celement); */
                       index=selement>>8;/* divide by 256 */
/* putfmt("A cluster has already occured here, index= %s.\n",index); */
                       for(t=0;(col[index][t]!=0)&&(t<10);t++){
                          sequence[t]=col[index][t];
                                                              }
                          sequence[t]=cluscode;
                       /* Test for existance of sequence */
                       for(n=0,found=0;(n<colindex)&&(found !=1);n++){
                          for(tt=0,count=0;tt<t;tt++){
putfmt("Comparing collision sequences, sequence= %c, col[]= %c.\n",sequence[tt],col[index][tt]);
                            if(sequence[tt]==col[index][tt]){
                              count++;
                                                            }
                                                          }
                        if(count==tt){ /* Sequence found */
                          putfmt("Sequence found, n= %s.\n",n);
                          *(MAP+totadd)=(256*n)+collid;
                          found=1;
                                     }
                                                         }
                      if(found !=1){ /* Sequence not found */
                        putfmt("Sequence not found.\n");
                        colindex++;
                        if(colindex >=256){
                          putfmt("Maximum number of colisions reached.\n");
                          putfmt("Mapping terminated too many clashes.\n");
                          goto toomany;
                                          }
                        for(tt=0;tt<t;tt++){
                          col[colindex][tt]=sequence[tt];
                                           }
                        putfmt("New cluster created, colindex= %c.\n",colindex);
                        *(MAP+totadd)=colindex*256+collid;
                                    }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
putfmt("Mapping operation complete. Number of clashes is %s.\n",colindex);
for(t=0;col[t][0]!=0;t++){
   putfmt("Collision index is %s. Clusters colliding are:- \n",t);
   for(tt=0;col[t][tt]!=0;tt++){
      putfmt(" %hc,",col[t][tt]);
                               }
      putfmt("\n");
                          }
toomany:
if(PRINTSET==1){
putfmt("\033P\n");/* Restore pica sized printing */
               }
putfmt("\n");
}
```

Figure AD45 Continued.

Figure AD46


```
#include<std.h>
maptest()
{
GLOBAL unsigned short MAP[0x40000];
GLOBAL unsigned char col[256][10];
GLOBAL char iname[8][10];
GLOBAL short npoints[8][20];
GLOBAL short ptlist[8][20][500];
GLOBAL char ru[8];
GLOBAL short cenU[8][20][3];
GLOBAL short minfe[4];
GLOBAL float div[4];
GLOBAL short PRINTSET;
unsigned char fmask,cmask;
unsigned int totadc,colindex;
unsigned short acdcmp,addmnx;
unsigned short addmnd,addmdx;
short cmp,mnx,mnd,mdx,obj,np;
unsigned short celement,collid,cminang,cmaxang;
unsigned char file,cluster,l,opt;
int n;
PRINTSET=0;
cmask=0x1f;
fmask=0xe0;
collid=0xff;
putfmt("Routine to test the cluster mapping.\n\n");
for(l=0;l !=1;){
putfmt("Enter four features corresponding to an object in the \n");
putfmt("following order: compaction, minmax, minmid, midmax.\n");
getfmt("%s %s %s %s",&cmp,&mnx,&mnd,&mdx);
putfmt("Div 0-3= %0.2f, %0.2f, %0.2f, %0.2f.\n",div[0],div[1],div[2],div[3]);
putfmt("cmp= %s, mnx= %s, mnd= %s, mdx= %s.\n",cmp,mnx,mnd,mdx);
addcmp=(short)((float)(cmp-minfe[0])/div[0]);
addmnx=((short)((float)(mnx-minfe[1])/div[1]))<<5;
addmnd=((short)((float)(mnd-minfe[2])/div[2]))<<10;
addmdx=((short)((float)(mdx-minfe[3])/div[3]))<<14;
putfmt("addcmp= %s, addmnx= %s, addmnd= %s, addmdx= %s.\n",acdcmp,addmnx,aodmnd,adcmax);
totadd=addcmp+addmnx+addmnd+addmdx;
putfmt("Address gereratec is %hi.\n",totadd);
celement=(*(MAP+totadd));
putfmt("Element is %hs.\n",celement);
if(celement==0){
  putfmt("No match found for object.\n");
                }
else if((celement & collid)!=collid){
       file=((celement & fmask)>>5);
       cluster=(celement & cmask);
      putfmt("The object corresponds to file %hc, cluster %c.\n",file,cluster);
       putfmt("The file description is %c.\n",&iname[file][0]);
       putfmt("The number of test images in the object cluster this belongs to is:- %s.\n"
,npoints[file][cluster]);
       putfmt("These points are:-");
       for(np=0;np<rpoints[file][cluster];np++){
          putfmt("%s, ",ptlist[file][cluster][np]);
                                                }
       putfmt("\n");
```

Figure AD46 Continued.

```
        cminang=(short)((float)(ru[file])*(float)ptlist[file][cluster][0]*0.9);
        cmaxang=(short)((float)(ru[file])*(float)ptlist[file][cluster][np-1]*0.9);
        putfmt("\n These points correspond to angles of orientation from %s to %s degrees.\n"
,cminang,cmaxang);
    putfmt("The distances of this centre from the three end points are: %s, %s, %s.\n"
,cenD[file][cluster][0],cenD[file][cluster][1],cenD[file][cluster][2]);
                                    }
else if((element & collid)==collid){
       putfmt("A cluster collision has occured at this point.\n");
       colindex=(element >> 8);
       putfmt("Collision index is %i.  Clusters colliding are:-\n",colindex);
       for(n=0;col[colindex][n]!=0;n++){
          file=((col[colindex][n] & fmask)>>5);
          cluster=(col[colindex][n] & cmask);
          putfmt("file %hc, cluster %c.\n",file,cluster);
       putfmt("The file description is %p.\n",&frame[file][0]);
       putfmt("The number of test images in the object cluster this belongs to is:- %s.\n"
,npoints[file][cluster]);
       putfmt("These points are:-");
       for(np=0;np<npoints[file][cluster];np++){
          putfmt("%s, ",ptlist[file][cluster][np]);
                                                 }
       putfmt("\n");
       cminang=(short)((float)(ru[file])*(float)ptlist[file][cluster][0]*0.9);
      cmaxang=(short)((float)(ru[file])*(float)ptlist[file][cluster][np-1]*0.9);
       putfmt("These points correspond to angles of orientation from %s to %s degrees.\n"
,cminang,cmaxang);
      putfmt("The distances of this centre from the three end points are: %s, %s, %s.\n"
,cenD[file][cluster][0],cenD[file][cluster][1],cenD[file][cluster][2]);
                                    }
                                       }
else {
       putfmt("Error in object identification.\n");
     }
putfmt("Do you want to test again ?\n");
if(getch()=='n'){
i=1;
                 }
getch();getch();
     }
}
```

Figure AD47

```
Feb 15 10:20:11 1988  arcorrect.c Page 1

#include<std.h>
arcorrect()
{
GLOBAL short cimpoint[100][25];
GLOBAL short totarea[100],bound[100];
GLOBAL short LINE,ascale;
register short q,n;
ascale=119;
putfmt("Area correction routine entered.\n");
r=(LINE-1);
for(q=0;q<n;q++){
   if((totarea[q] !=0) && (cimpcint[q][0] !=0)){
     totarea[q] += ((cimpoint[c][0]-145)*cimpcint[c][19])/ascale;
                                                 }
                        }
}
```

Figure AD48

```
#include<std.h>
identify()
{
GLOBAL short HCRES;
GLOBAL short paction[100];
GLOBAL short bound[100];
GLOBAL short cimpcint[100][25];
GLOBAL short lilen,LINE;
GLOBAL unsigned short MAP[0x40000];
GLOBAL unsigned char col[256][10];
GLOBAL char iname[8][10];
GLOBAL short npoints[8][20];
GLOBAL short ptlist[8][20][500];
GLOBAL char ru[8];
GLOBAL short cenD[8][20][3];
GLOBAL short minfe[4],maxfe[4],fetol[4];
GLOBAL float div[4];
GLOBAL unsigned char afile[100][5],acluster[100][5];
register short q,t;
register int n;
unsigned char fmask,cmask;
unsigned int totadd,colindex;
unsigned short acdcmp,addmnx;
unsigned int addmnd,addmdx;
short cmp,mnx,mnd,mdx,obj,np,set;
unsigned short celement,collid,cminang,cmaxang;
unsigned char file,cluster,l,opt;
cmask=0x1f;
fmask=0xe0;
collid=0xff;
if(HCRES==1){ /* Set elite print size */
  putfmt("\033M");
            }
for(q=0;q<100;q++){
   for(n=0;n<5;n++){
      afile[q][n]=0;
      acluster[q][n]=0;
                    }
                   }
putfmt("\nObject identification by mapping into the look-up table.\n\n");
/* putfmt("Feature scaling factors are: (Div 0-3)= %0.2f, %0.2f, %0.2f, %0.2f.\r\n"
,div[0],div[1],div[2],div[3]); */
for(q=0;q<(LINE-1);q++){
   if((bound[q]>lilen)&&(cimpoint[q][19]>35)){
     putfmt("Testing data set %s.\n",q);
   for(t=0,set=0;t<9;t +=3){
      if((cimpoint[q][t]==0)||(cimpoint[q][t+1]==0)){
        set= -1;
                                                    }
                          }
        if(set==1){
        putfmt("Data set %s ignored because of zero alternative points.\n",c);
                  }
    if(paction[q]==0){
      set= -1;
    putfmt("Data set %s ignored for because of zero compaction feature.\n",c);
```



```
                     }
    if((paction[q]<(minfe[0]-fetol[0])) || (paction[q]>(maxfe[0]+fetol[0]))){
      set = -1;
      putfmt("Compaction feature for data set %s is out of range.\n",q);
                                                                       }
     if((cimpoint[q][17]<(minfe[1]-fetol[1])) || (cimpcint[q][17]>(maxfe[1]+fetol[1]))){
      set = -1;
      putfmt("Minmax feature for data set %s is out of range.\n",q);
                                                                   }
     if((cimpoint[q][15]<(minfe[2]-fetol[2])) || (cimpcint[q][15]>(maxfe[2]+fetol[2]))){
      set = -1;
      putfmt("Minmid feature for data set %s is out of range.\n",q);
                                                                   }
     if((cimpoint[q][16]<(minfe[3]-fetol[3])) || (cimpcint[q][16]>(maxfe[3]+fetol[3]))){
      set = -1;
      putfmt("Midmax feature for data set %s is out of range.\n",q);
                                                                   }
putfmt("\n");
if(set != -1){
  if(paction[q]<0){
    cmp=(-paction[q]);
                  }
```

Figure AD48 Continued.

```
  else {
       cmp=paction[q];
       }
  rnx=cimpoint[q][17];
  rnd=cimpoint[q][15];
  mdx=cimpoint[q][16];
putfmt("The features for object %s are:- cmp= %s, rnx= %s, rnd= %s, mdx= %s.\n",c,cmp,rnx,rnd,mdx);
addcmp=(short)((float)(cmp-minfe[0])/div[0]);
addrnx=((short)((float)(rnx-minfe[1])/div[1]))<<5;
addrnd=((short)((float)(rnd-minfe[2])/div[2]))<<10;
addmdx=((int)((float)(mdx-minfe[3])/div[3]))<<14;
/* putfmt("Scaled addresses generated are:- acccmp= %hs, addmnx= %hs, addmnd= %hs, addmdx= %hi.\n"
,acddcmp,addmnx,addmnd,addmdx); */
totadd=addcmp+addmnx+addmnd+addmdx;
/* putfmt("Complete Look-up table address generated is %hi (%i).\n",totacd,totadc); */
celement=(*(MAP+totacd));
putfmt("Element in Look-up table is %hs.\n",celement);
putfmt("\n");
if(celement==0){
  putfmt("No match with learnt objects found for this object.\n\n");
                }
else if((celement & collid)!=collid){
       afile[q][0]=file=(((celement & fmask)>>5)-1);
       afile[q][0] +=1;
       acluster[q][0]=cluster=(celement & cmask);
      putfmt("This object corresponds to file %hc, cluster %c.\n",file,cluster);
       putfmt("The description of this file is \"%p.\"\n",&iname[file][0]);
       putfmt("The number of test images in the object cluster this belongs tc is:- %s.\r"
,npcints[file][cluster]);
/*       putfmt("The indices of these test images are:-");
       for(np=0;np<npoints[file][cluster];np++){
          putfmt("%s, ",ptlist[file][cluster][np]);
          if(((np+1) % 12) ==0)putfmt("\n");
                                              } */
```



```
      np=npoints[file][cluster];
       cminang=(short)((float)(ru[file])*(float)ptlist[file][cluster][0]*0.9);
       cmaxang=(short)((float)(ru[file])*(float)ptlist[file][cluster][np-1]*0.9);
       putfmt("\nThese points correspond to angles of orientation from %s to %s degrees.\r",cminang,cmaxang);
    putfmt("The distances of this object's centre from the three end points are: %s, %s, %s.\r"
,cenD[file][cluster][0],cenD[file][cluster][1],cenD[file][cluster][2]);
    putfmt("\n");
                                     }
else if((celement & collid)==collid){
       putfmt("A cluster collision has occured at this point.\n");
       colindex=(celement >> 9);
       putfmt("The collision index of this collision is %i. The clusters colliding are:-\r\n",colindex);
       for(n=0;col[colindex][n]!=0;n++){
          afile[q][n]=file=(((col[colindex][n] & fmask)>>5)-1);
          afile[q][n] +=1;
          acluster[q][n]=cluster=(col[colindex][n] & cmask);
          putfmt("file %hc, cluster %c.\n",file,cluster);
       putfmt("The file description is %p.\r",&iname[file][0]);
       putfmt("The number of test images in the object cluster this belongs tc is:- %s.\r"
,npoints[file][cluster]);
/*       putfmt("These points are:-");
       for(np=0;np<npoints[file][cluster];np++){
          putfmt("%s, ",ptlist[file][cluster][np]);
          if(((np+1) % 12) ==0)putfmt("\n");
                                                 }
       putfmt("\n"); */
       no=npoints[file][cluster];
        cminang=(short)((float)(ru[file])*(float)ptlist[file][cluster][0]*0.9);
       cmaxang=(short)((float)(ru[file])*(float)ptlist[file][cluster][np-1]*0.9);
putfmt("These points correspond to angles of orientation from %s to %s degrees.\r",cminang,cmaxang);
       putfmt("The distances of the pick-up point from the three end points are: %s, %s, %s.\r\r"
,cenD[file][cluster][0],cenD[file][cluster][1],cenD[file][cluster][2]);
                                              }
    putfmt("\n");
                                                 }
else {
       putfmt("Error in object identification.\n");
     }
               }
              }
          }
if(HCRES==1){ /* Reset pica print size */
  putfmt("\033P\n");
            }
}
```

Figure AD49

```
#include<std.h>
circent()
{
GLOBAL short cenD[8][20][3];
GLOBAL short cimpoint[100][25];
GLOBAL short bound[100];
GLOBAL short paction[100];
GLOBAL short LINE,lilen;
GLOBAL short cross[10][4][2][14];
GLOBAL short crossind;
GLOBAL unsigned char afile[100][5],acluster[100][5];
short tick,n,m,run,set,t;
register short q,R1,R2;
unsigned char file,cluster;
short x,y,Y1,Y2,X1,X2;
double A,B,a,b,c,d,xf,yf;
double sqrt();
crossind=0;
tick=10;
putfmt("Routine to determine intercept points entered.\n");
for(q=0;q<(LINE-1);q++){
   if(afile[q][0]!=0){
  file=(afile[q][0]-1);
  cluster=acluster[q][0];
  putfmt("Intercept point calculation for file= %c, cluster= %c.\n",file,cluster);
  for(run=0;run<3;run++){
     if(run==0){
       R1=cenD[file][cluster][0];
       R2=cenD[file][cluster][1];
/*       putfmt("R1= %s, R2= %s.\n",R1,R2); */
       X1=cimpoint[q][0];
       X2=cimpoint[q][3];
       Y1=cimpoint[q][1];
       Y2=cimpoint[q][4];
                }
     if(run==1){
       R1=cenD[file][cluster][0];
       R2=cenD[file][cluster][2];
/*       putfmt("R1= %s, R2= %s.\n",R1,R2); */
       X1=cimpoint[q][0];
       X2=cimpoint[q][6];
       Y1=cimpoint[q][1];
       Y2=cimpoint[q][7];
                }
     if(run==2){
       R1=cenD[file][cluster][1];
       R2=cenD[file][cluster][2];
       X1=cimpoint[q][3];
       X2=cimpoint[q][6];
       Y1=cimpoint[q][4];
       Y2=cimpoint[q][7];
                }
A=(double)(((R1+R2)*(R1-R2))-((Y1+Y2)*(Y1-Y2))-((X1+X2)*(X1-X2)))/(double)(2*(Y2-Y1));
B=(double)((double)(X1-X2)/(double)(Y2-Y1));
a=(1+(B*B));
b=(2*((A*B)-(B*Y1)-(X1))); /* negative ignored */
```



```
c=(A*A)-(2*(A*Y1))+(Y1*Y1)+(X1*X1)-(R1*R1);
c=((b*b)-(4*a*c));
if(c<0){
  d=0;
/*  putfmt("Single root solution to intersect problem.\n"); */
       }
/* putfmt("R1= %s, R2= %s, X1= %s, X2= %s,Y1= %s, Y2= %s, A= %C.4f, B= %C.4f.\n"
,R1,R2,X1,X2,Y1,Y2,A,B); */
/* putfmt("Coefficients are: a=%0.4f, b=%0.4f, c=%0.4f, c=%0.4f.\n",a,b,c,d); */
xf=(-b+(double)(sqrt((double)(d))))/(2*a); /* Positive root soln */
yf=A+(B*xf);
x=(short)xf;
y=(short)yf;
```

```
/* putfmt("From positive square root for run=%s is: x=%s, y=%s.\n",run,x,y); */
cross[crossind][run][0][0]=x;
cross[crossind][run][0][1]=y;
cross[crossind][run][0][2]=x-tick;
cross[crossind][run][0][3]=y-tick;
cross[crossind][run][0][4]=x+tick;
cross[crossind][run][0][5]=y+tick;
cross[crossind][run][0][6]=x;
cross[crossind][run][0][7]=y;
cross[crossind][run][0][8]=x+tick;
cross[crossind][run][0][9]=y-tick;
cross[crossind][run][0][10]=x-tick;
cross[crossind][run][0][11]=y+tick;
cross[crossind][run][0][12]=x;
cross[crossind][run][0][13]=y;
xf=(-b-(double)(sqrt((double)(d))))/(2*a); /* Negative root soln */
yf=A+(B*xf);
x=(short)xf;
y=(short)yf;
/* putfmt("From negative square root for run=%s is: x=%s, y=%s.\n",run,x,y); */
cross[crossind][run][1][0]=x;
cross[crossind][run][1][1]=y;
cross[crossind][run][1][2]=x-tick;
cross[crossind][run][1][3]=y-tick;
cross[crossind][run][1][4]=x+tick;
cross[crossind][run][1][5]=y+tick;
cross[crossind][run][1][6]=x;
cross[crossind][run][1][7]=y;
cross[crossind][run][1][8]=x+tick;
cross[crossind][run][1][9]=y-tick;
cross[crossind][run][1][10]=x-tick;
cross[crossind][run][1][11]=y+tick;
cross[crossind][run][1][12]=x;
cross[crossind][run][1][13]=y;
}
crossind++;
}}
}
```

Figure AD49 Continued.

Feb 15 10:21:13 1988 cockhat.c Page 1

Figure AD50

```
#include<std.h>
cockhat()
{
GLOBAL short cross[10][4][2][14];
GLOBAL short crossind;
short midncx,midxcx,micmaxx0,midmaxx1,lmnx,xcen;
short midncy,micxcy,midmaxy0,midmaxy1,lmny,ycen;
short n,tick,d1,d2;
register short cr,mcx,mcy;
tick=10;
putfmt("Intersect triangle centre of gravity routine entered.\n");
for(cr=0;cr<crossind;cr++){
   midncx=cross[cr][0][1][0];
   midncy=cross[cr][0][1][1];
   midxcx=cross[cr][2][1][0];
   micxcy=cross[cr][2][1][1];
   mcx=(micncx+midxcx)/2; /* midpoints of two min arc intersects */
   mcy=(micncy+micxcy)/2;
   midmaxx0=cross[cr][1][0][0];
   midmaxy0=cross[cr][1][0][1];
   midmaxx1=cross[cr][1][1][0];
   midmaxy1=cross[cr][1][1][1];
   d1=(((midmaxx0-mcx)*(midmaxx0-mcx))+((midmaxy0-mcy)*(midmaxy0-mcy)));
   d2=(((midmaxx1-mcx)*(midmaxx1-mcx))+((midmaxy1-mcy)*(midmaxy1-mcy)));
   if(d1<d2){
     lmnx=midmaxx0;
     lmny=midmaxy0;
           }
   else {
     lmnx=midmaxx1;
     lmny=midmaxy1;
        }
   xcen=lmnx+(((mcx-lmnx)*2)/3);
   ycen=lmny+(((mcy-lmny)*2)/3);
 putfmt("For cross index=%s, x centre= %s, y centre= %s.\n",cr,xcen,ycen);
   cross[cr][3][0][0]=xcen;
   cross[cr][3][0][1]=ycen;
   cross[cr][3][0][2]=xcen-tick;
   cross[cr][3][0][3]=ycen-tick;
   cross[cr][3][0][4]=xcen+tick;
   cross[cr][3][0][5]=ycen+tick;
   cross[cr][3][0][6]=xcen;
   cross[cr][3][0][7]=ycen;
   cross[cr][3][0][8]=xcen+tick;
   cross[cr][3][0][9]=ycen-tick;
   cross[cr][3][0][10]=xcen-tick;
   cross[cr][3][0][11]=ycen+tick;
   cross[cr][3][0][12]=xcen;
   cross[cr][3][0][13]=ycen;
                        }
}
```

Figure AD51

```
Feb 15 10:22:21 1988  cache.s Page 1

*       Program to start cache running on the 68020
*
        .globl  _cache
*
*************************************************************
_cache:
        move.l  #1,d0
        .word   0x4e7b,2
        rts
*
*       End of routine
```

Figure AD52

```
Feb 15 10:22:21 1988  duart.s Page 1

*       Functions for serial o/p using on-board 68020 duart.
*       Call  initdu() to initialize both sides.
*            puts2() for o/p from duart 2.
*            puts3() for o/p from duart 3.
*
*************************************************************

        .text
        .even
        duart2=0xffe00020
        .globl _initdu
        .globl _puts2
        .globl _puts3

_initdu:
        move.l  #duart2,a0
        move.b  #0x30,4(a0)

        move.l  #0,5(a0)
        bsr     slin
        add.w   #8,a0
        bsr     slin
        move.b  #3,duart2+14
        rts

slin:
        move.b  #0x2a,2(a0)
        move.b  #0x3a,2(a0)
        move.b  #0x1a,2(a0)

        move.b  #0x93,(a0)

        move.b  #0x17,(a0)

        move.b  #0xbb,1(a0)
        move.b  #0x05,2(a0)
        rts

_puts2:
        move.l  #duart2,a0
loop1:
        btst    #2,1(a0)
        beq.s   loop1
        move.b  d0,3(a0)
        rts

_puts3:
        move.l  #duart2+8,a0
loop2:
        btst    #2,1(a0)
        beq.s   loop2
        move.b  d0,3(a0)
        rts

*       End of routines.
```

Figure AD53

```
#include <stdio.h>
pripw() /* procuce correct bit formations for the dot matrix printer */
        /* from the expanded array and output through the serial port*/
        /* to the printer */
{
unsigned short i,j,p,m,mask,t,d;
static char title[]=" Print of current image being worked on.\n";
short fd;
char buf[25];
register char *pmpixel;
GLOBAL char mpixel[53761];
/* autoen(1); */ /* Enable serial handshake */
m=0;
pmpixel=mpixel;
buf[1]=0x1b;/* printer control codes */
buf[2]=0x41;
buf[3]=0x08;
buf[4]=0x1b;/* ESC */
buf[5]=0x4b;
buf[6]=0x18;
buf[7]=0x01;
buf[8]=0x0a;
buf[9]=0xff;
buf[10]=0x01;
buf[11]=0x80;
buf[12]=0x1a;
buf[13]=0x0d;
buf[14]=0x0a;
buf[15]=' ';
buf[16]=0xCf;/* Condense */
buf[17]='W';/* Enlarge */
buf[18]=0x01;
buf[19]=0x12;
buf[20]='0';
        hwrite2(&buf[4],1);
        hwrite2(&buf[16],1);
        hwrite2(&buf[4],1);
        hwrite2(&buf[17],1);
        hwrite2(&buf[19],1);
              for(i=0;i<5;i++){
                 hwrite2(&buf[15],1);
                              }
                    hwrite2(&buf[13],2);/* set up printer for dot image mode */
                    hwrite2(&buf[13],2);
                    hwrite2(&buf[1],3);
                    hwrite2(&buf[4],2);
                    hwrite2(&buf[12],1);
                    hwrite2(&buf[7],1);
                    for(i=0;i<282;i++){
                       hwrite2(&buf[10],1);
                                       }
                       hwrite2(&buf[13],2);
                 while(m<185){
                      hwrite2(&buf[1],3);
                      hwrite2(&buf[4],2);
                      hwrite2(&buf[12],1);
                      hwrite2(&buf[7],1);
                        hwrite2(&buf[9],1);
for(i=0;i<29C;i++){/* vertical bit pattern formation for dot matrix printer head */
   for (j=m,mask=128,p=0;j<m+8;j++){
       t=i+j*28C;
       if(*(pmpixel+t)==1){
         d=mask*(*(pmpixel+t));
         p=p+d;
                          }
       mask >>=1;
                                  }
   buf[0]=p;
   hwrite2(&buf,1);
                }
   hwrite2(&buf[9],1);
   hwrite2(&buf[8],1);
   m=m+8;
                }
                hwrite2(&buf[1],3);
                hwrite2(&buf[4],2);
                hwrite2(&buf[12],1);
                hwrite2(&buf[7],1);
                for(i=0;i<282;i++){
                   hwrite2(&buf[11],1);
                                   }
                    hwrite2(&buf[13],2);
                    hwrite2(&buf[13],2);
                  for(i=0;title[i] !=0;i++){
                    hwrite2(&title[i],1); /* write out the title */
                                           }
                    hwrite2(&buf[13],2);
hwrite2(&buf[13],2); /* reseting of printer */
hwrite2(&buf[19],1);
hwrite2(&buf[4],1);
hwrite2(&buf[17],1);
hwrite2(&buf[20],1);
hwrite2(&buf[4],1);
```

Figure AD54


```
#include <std.h>
printable() /* feature table printing routine */
{
GLOBAL short cimpcint[100][25];
GLOBAL short LINE;
GLOBAL int pirdex;
GLOBAL short bound[100];
GLOBAL short totperi[100];
GLOBAL short totarea[100];
GLOBAL short paction[100];
GLOBAL short lilen;
BYTES itob();
BYTES decode();
shcrt t,n,bl;
int m;
char util[6];
char xuf[6];
static char buf[]="          TABLE of IMAGE FEATURES \n\n";
static char euf[]="-------------------------------------------------------------------------------
static char huf[]="|      |                 |                 |                 |       |       |       |
static char fuf[]="|      |   minpcint      |   midpcint      |    maxpcint     |       |       |       |
static char guf[]="|Sample|  i    j    pid |  i    j    pid |  i    j    pid | Bcund | Perim | Area  |Cmpactic
char cuf[5];
char duf[15];
util[0]='!';/* control codes etc for Epson printer */
util[1]=' ';
cuf[0]=0x0e;/* enlarge */
cuf[1]=0x45;/* emphasise */
cuf[2]=0xCd;/* CR */
cuf[3]=0x0a;/* LF */
cuf[4]=0x1b;/* ESC */
cuf[5]=0x0f;/* Concensed */
cuf[6]=0x12;/* con off */
cuf[7]='M';/*  Elite */
cuf[8]='P';/*  Pica  */
cuf[9]='0';/*  0 default spacing */
hwrite2(&duf[4],1);
hwrite2(&duf[9],1);/* reset cefaults */
hwrite2(&duf[0],1);
for(t=0;buf[t]!=C;t++){
   hwrite2(&buf[t],1);
                      }
hwrite2(&duf[4],1);
hwrite2(&duf[7],1);/* Elite */
hwrite2(&duf[4],1);
hwrite2(&duf[9],1);/* cefault spacing  */
for(t=0;euf[t]!=0;t++){
   hwrite2(&euf[t],1);
                      }
for(t=0;huf[t]!=C;t++){
   hwrite2(&huf[t],1);
                      }
for(t=0;fuf[t]!=C;t++){
   hwrite2(&fuf[t],1);
                      }
for(t=0;guf[t]!=C;t++){

   hwrite2(&guf[t],1);
                      }
for(t=0;huf[t]!=0;t++){
   hwrite2(&huf[t],1);
                      }
for(t=0;euf[t]!=0;t++){
   hwrite2(&euf[t],1);
                      }
for(n=0;n<(LINE-1);n++){
      if(bound[n]>lilen){
```

Figure AD54 Continued.

```
/* Meat of program : value output */
hwrite2(&util,1);/* send '|' */
hwrite2(&util[1],1);/* send ' ' */
cecode(&xuf,6,"%- 6s",n);
hwrite2(&xuf,5);
t=0;
for(bl=0;bl<3;bl++){
   hwrite2(&util,1);
   hwrite2(&util[1],1);
   decode(&xuf,6,"%- 6s",cimpoint[n][t]);
   hwrite2(&xuf,5);
   t++;
   decode(&xuf,6,"%- 6s",cimpoint[n][t]);
   hwrite2(&xuf,5);
   t++;
   decode(&xuf,6,"%- 6s",cimpoint[n][t]);
   hwrite2(&xuf,3);
   t++;
                    }
hwrite2(&util,2);
decode(&xuf,6,"%- 6s",bound[n]/4);
hwrite2(&xuf,6);
hwrite2(&util,2);
decode(&xuf,6,"%- 6s",totperi[r]);
hwrite2(&xuf,6);
hwrite2(&util,2);
decode(&xuf,6,"%- 6s",totarea[n]);
hwrite2(&xuf,6);
        hwrite2(&util,2);
        decode(&xuf,6,"%- 6s",paction[r]);
        hwrite2(&xuf,6);
    /*    hwrite2(&util,2);
        decode(&xuf,6,"%- 6s",cimpcint[r][11]);
        hwrite2(&xuf,6);
        hwrite2(&util,2);
        decode(&xuf,6,"%- 6s",cimpcint[n][14]);
        hwrite2(&xuf,6); */
        for(t=15;t<18;t++){
           hwrite2(&util,2);
           hwrite2(&util[1],1);
           decode(&xuf,6,"%- 6s",cimpcint[n][t]);
           hwrite2(&xuf,6);
                          }
        decode(&xuf,6,"%- 6s",cimpcint[n][19]);/*senc total mir mid max cist */
        hwrite2(&util,2);
        hwrite2(&xuf,4);
        hwrite2(&util,1);
        hwrite2(&duf[2],2);
        hwrite2(&duf[3],1);
          }
         }
/* End of printing icop */
        for(t=0;huf[t]!=0;t++){
           hwrite2(&huf[t],1);
                              }
        for(t=0;euf[t]!=0;t++){
           hwrite2(&euf[t],1);
                              }
        hwrite2(&duf[4],1);
        hwrite2(&duf[9],1);
}
```

Figure AD55


```
#include <std.h>
shipccrnpw()/* to cutput line corners for plotting on CLET  */
/* note hwrite3 sends 8 bits sc two are usec tc senc a short MSByte first*/
{
GLCBAL short *line[1C0];
GLCBAL short LINE,BP,fd;
GLCBAL short bcund[1C0];
register short n;
register short *pocint,*pbuf;
unsigned short buf[7];
short m,endchar;
int c;
short k=0;
enochar=0xffff;
buf[3]=0x58;/* start character X */
buf[4]=0xfe;/* stop character */
buf[5]=0x41;/* start of sequence character A */
buf[6]=0x90;/* end nf secuence character (144) */
putfmt("Corner outputting routine entered.\n\n");
hwrite3(&buf[5],2); /* send start of seclence char */
       for(m=0;m<(LINE-1);m++){
          if(bcund[m]>20){
               hwrite3(&buf[3],2);/* send start character */
               ppcint=line[m];/* puts line starting pcirt in buf array */
               buf[0]=(*(npoint++));
               if(buf[0]==254){ buf[0]=253;}
               buf[1]=(*(npcint++));
               if(buf[1]==254){ buf[1]=253; }
               cpoint++; ppcint++;
   /* putfmt("line[m]= %i pncint= %i *pccirt= %s \n",line[m],ppcint,*ppcint);*/
               hwrite3(&buf[0],4);/*send first character */
               while(*pncint !=encchar){
                    pbuf=buf;
                    if (*(ppcint+3)>0){
                       k++;
                       *(pbuf++)=(*(ppcint++));
                    if(buf[0]==254){ buf[0]=253; }
                    *(pbuf++)=(*(ppcint++));
                    if(buf[1]==254){ buf[1]=253; }
                    ppcint++;/*inc tc ignore dir element */
                    ppcint++;/*inc tc ignore s element */
                    hwrite3(&buf[0],4);/* output x,y cocrds as four bytes */
/*                  putfmt("points being output= %s, %s ,%s\r",buf[0],buf[1],k); */
                    for(c=0;c<10000;c++)()/* wait lccp fcr CLET */
                                 }
                    else
                        cpoint=cpoint+4;
                                  }
               buf[0]=(*(npoint-4));/* build up final character */
               if(buff[0]==254){ buf[0]=253; }
                 buf[1]=(*(cpoint-3));
               if(buf[2]==254){ buf[2]=253; }
               hwrite3(&buf[0],4);/* senc final character */
               hwrite3(&buf[4],2);/* senc stop character */
                    for(c=0;c<5000C;c++)()/* wait lccp fcr CLET */
                             }
```



```
                                  }
hwrite3(&buf[6],2); /* send end of secuerce character */
putfmt("End of secuence marker sent,-entering wait state.\n");
for(c=0;c<10CCC0C;c++)()/* wait locp for DLET */
         putfmt(" Corner transfer completec \n");
}
```

Figure AD56


```
#include<std.h>
shipcen()
{
GLOBAL short EOS,PLOTCORN,cross[10][4][2][14];
GLOBAL short crossind;
short t,ind,interc,bin,corn;
unsigned short buf,sbuf,ebuf,eosbuf,sosbuf;
sosbuf=0x41; /* Start of sequence char A */
sbuf=0x58;   /* Start character X */
ebuf=0xfe;   /* Stop char */
eosbuf=144; /* End of plot sequence marker */
putfmt("Into shipcen routine.\n");
hwrite3(&sosbuf,2); /* Start of sequence buffer */
      for(ind=0;ind<crossind;ind++){
         for(interc=3;interc<4;interc++){
            putfmt("Values for intercept %s are:-",interc);
            for(bin=0;bin<2;bin++){
                hwrite3(&sbuf,2);
            for(corn=0;corn<14;corn++){
               buf=cross[ind][interc][bin][corn];
               if(buf==254)buf=253;
               for(t=0;t<10000;t++){} /* Wait loop for DLET */
               putfmt("%s, ",buf);
               hwrite3(&buf,2);
                                    }
            outfmt("Centre set %s output, centre=%s.\n",ind,interc);
            hwrite3(&ebuf,2);
                  for(t=0;t<90;t++){ /* Wait loop for DLET */
                     putfmt(".");
                                    }
                  outfmt("\n");
                  if(interc==3){ /* only output a single centre point */
                    putfmt("Incrementing bin.\n");
                    bin++;
                               }
                                  }
                                     }
                                  }
            for(t=0;t<10000;t++){} /* Wait loop for DLET */
            if(EOS==1){ /* Send end of plot sequence marker */
              hwrite3(&eosbuf,2);
              putfmt("End of plot sequence marker sent.\n");
                      }
            if(PLOTCORN==1){
              putfmt("Press carriage return to continue.\n");
              getch();getch();
                           }
}
```

Figure ADU1

```
5 REM Program to read in a series of line corner coordinates
6 REM from the DUMC M68000 Unit.
7 REM Lines are plotted between these points on the DUET screen and
8 REM the corners stored as coordinates in a specified DUET file.
10 PRINT"Enter name of file in which image is to be stored."
15 INPUT NM$
20 REM Open chosen storage file.
25 OPEN NM$ FOR OUTPUT AS #3
30 REM Open serial communication line.
35 OPEN"COM1:9600,N,8,1,BIN"AS #1
40 REM Interrupt driven file closing routine.
45 ON KEY(6) GOSUB 315
50 DIM PT(2,1500)
55 COLOR 0,5,0
60 NN=1
65 B=254
70 REM Main program loop.
75 FOR N=1 TO 500
80 KEY(6) ON
85 PRINT "N= "N
90 A$=INPUT$(1,#1)
95 KEY (6) OFF
100 REM Test for start character.
105 IF A$<>"X" THEN GOTO 90
110 FOR Y=0 TO 500
115 REM I=0 For X coordinate.
120 REM I=1 For Y coordinate.
125 FOR I=0 TO 1
130 A$=INPUT$(1,#1)
135 A=ASC(A$)
140 REM Test for end character.
145 REM On end character go to draw routine.
150 IF A=254 THEN PRINT #3,B:GOTO 215
155 REM Take MSB
160 YH=256*A
165 A$=INPUT$(1,#1)
170 REM Take LSB
175 YL=ASC(A$)
180 IF YL=254 THEN PRINT #3,B:GOTO 215
185 YY=YH+YL
190 PRINT #3,YY
195 PT(I,Y)=YY
200 NEXT I
205 NEXT Y
210 STOP
```

Figure ADU1 Continued.

```
215 REM Drawing single shape subroutine
220 PRINT"DRAWING"
225 SCREEN 2
230 COLOR 1,NN,0
235 T=0:YP=Y-1
240 PX=2*PT(0,T)+50
245 PY=2*PT(1,T)
250 REM Move to first point on the outline.
255 PSET(PX,PY)
260 FOR T=1 TO YP
265 REM Scale points.
270 X=2*PT(0,T)+50:Z=2*PT(1,T)
275 REM Draw points.
280 LINE -(X,Z)
285 NEXT T
290 REM Colour setting variables.
295 NN=NN+1
300 IF NN=7 THEN NN=1
305 REM End of the main loop.
310 NEXT N
315 REM File closing subroutine
320 CLOSE #1
325 CLOSE #3
330 PRINT"The name of the file just saved is :-"NM$
335 STOP
340 RETURN
```

Figure ADU2

```
4 REM Program which first configures a Hewlett-Packard Plotter and the
6 REM reads in a set of line corner coordinates from a specified file.
8 REM These are then output in a suitable form to the plotter.
30 PRINT"Enter Name of Image-File to be plotted."
40 INPUT A$
50 B$=CHR$(3):REM End of text character
60 DIM PT(2,1500)
70 NN=3:C=3
80 REM Open serial communication line.
90 OPEN"COM1:9600,N,8,1,BIN"AS#1
100 REM Open specified input file.
110 OPEN A$ FOR INPUT AS #3
120 REM Scale,draw and label plotting area.
130 PRINT#1,"IN;SP1;IP1250,750,7750,7250;"
140 PRINT#1,"SC0,300,0,300;"
150 PRINT#1,"PU60,20PD360,20,360,220,60,220,60,20PU"
160 PRINT#1,"SI.2,.3TL1.5,0"
170 FOR X=60 TO 360 STEP 60
180 PRINT#1,"PA";X,",20;XT;"
190 XM=X-60
200 PRINT#1,"CP-2,-1;LB";XM;B$
210 NEXT X
220 PRINT#1,"PA66.5,20;CP17,-3;LBImage X Coordinate."B$
230 H=200
240 FOR Y=20 TO 220 STEP 40
250 PRINT#1,"PA60,",Y,"YT;"
260 IF Y>120 THEN PRINT#1,"CP-3,-.25;LB";H;B$
270 IF Y<121 THEN PRINT#1,"CP-4,-.25;LB";H;B$
280 H=H-40
290 NEXT Y
300 PRINT#1,"PA60,220CP-3.5,2LBImage Y Coordinate."B$
310 PRINT#1,"SF1;PA8,250SI.3,.4CP0.5,2.0"
320 PRINT#1,"LB PLOT GENERATED BY IMAGE PATTERN CORNERS"B$
330 REM Wait loops.
340 FOR L=0 TO 1600
350 FOR K=1 TO 10:NEXT K
360 PRINT"L= "L:NEXT L
370 REM Main line loop.
380 FOR T=0 TO 500
390 REM Main point loop.
400 FOR N=0 TO 500
410 REM Test for EOF.
420 IF EOF(3) THEN GOTO 850
430 INPUT #3,X
440 REM Test for line endchar.
450 IF X=254 THEN GOTO 510
460 PT(0,N)=X
470 INPUT #3,Y
480 IF Y=254 THEN GOTO 510
490 PT(1,N)=Y
500 NEXT N
510 REM Draw and plot routines.
```

Figure ADU2 Continued.

```
520 SCREEN 2
530 COLOR 1,NN,0
540 M=0
550 N=N-1
560 REM Set up first point on the line.
570 X=PT(0,0)
580 PRINT"X= "X
590 Y=PT(1,0)
600 PRINT"Y= "Y
610 XD=2*X:YD=2*Y
615 X=X+60:Y=Y-20
620 REM Move to line start point on screen.
630 PSET(XD,YD)
640 YP=(200-Y)
645 REM Move to line start point on plotter.
650 PRINT#1,"SP";C;"LT;PA";X;YP;"PD"
660 REM Main draw and plot routines.
670 FOR M=1 TO N
680 REM Wait loop.
690 FOR L=0 TO 100:NEXT L
700 X=PT(0,M)
710 Y=PT(1,M)
720 XD=2*X:YD=2*Y
730 X=X+60:Y=Y-20
740 YP=(200-Y)
745 REM Draw line to point.
750 LINE -(XD,YD)
755 REM Plot line to point.
760 PRINT#1,"PA";X;YP
770 NEXT M
780 PRINT#1,"PU"
790 NN=NN+1
800 C=C+1
810 IF NN=6 THEN NN=3
820 IF C=6 THEN C=3
830 NEXT T
840 STOP
850 REM File closing routine.
860 PRINT "File printed was :-"A$
870 CLOSE #3
880 CLOSE #1
890 STOP
```

Figure ADU3

```
£define startchar 0x11
£define endchar 0x13
£include <stdio.h>
£include <conio.h>
£include <dos.h>
union REGS inregs,outregs;
FILE *fopen(),*f2p;
char F2[6];
main() /* Program to read in and store feature data from 68020 system */
{
int fprintf();
unsigned int nf,n,t,ta[2000],*pta,c;
unsigned char a,b,d,*res;
printf("Extended feature data storage program running.\n\n");
printf("Enter name of file into which the data is to be stored.\n\n");
res=gets(F2);
printf("Features are being stored in file %s.\n\n",F2);
/* Serial line already open. Default identifier stdaux. */
f2p=fopen(F2,"w");
if(f2p==NULL){
  printf("Error in opening storage file %s.\n",F2);
             }
printf("About to test for start character.\n");
inregs.h.ah=3; /* Set up calls to BDOS char read routine */
outregs.h.al=0;
for(;(d=outregs.h.al) !=startchar;){
   intdos(&inregs,&outregs);
   if(ferror(stdaux)) {
     perror("Read Error. \n");
     clearerr(stdaux);
                      }
      if(feof(stdaux)) {
        printf("End of file read.\n");
                       }
   printf("Testing for start character. Value read = %d.\n",d);
                                       }
printf("About to read in a and b.\n");
intdos(&inregs,&outregs);
a=outregs.h.al;
intdos(&inregs,&outregs);
b=outregs.h.al;
/* printf("Value read in for a= %d,b= %d.\n",a,b); */
if(a==endchar ||b==endchar) goto end;
nf=256*a + b;
printf("Number of sets of data to be stored is :- %d. \n",nf);
for(t=0,pta=ta;t<nf;t++){ /* Read in feature data */
   for(n=0;n<=11;n++){
      intdos(&inregs,&outregs);
      a=outregs.h.al;
     if(feof(stdaux)) {
       printf("End of file read.\n");
                      }
      if(ferror(stdaux)) {
        perror("Read Error.\n");
        clearerr(stdaux);
                         }
      if(a==endchar) goto end;
      intdos(&inregs,&outregs);
      b=outregs.h.al;
      if(ferror(stdaux)) {
        perror("Read Error.\n");
        clearerr(stdaux);
                         }
       if(b==endchar) goto end;
       c=256*a+b;
       *(pta++)=c;
                      }
     printf("Received values are:- %d,%d,%d,%d,%d,",
*(pta-12),*(pta-11),*(pta-10),*(pta-9),*(pta-8));
      printf("%d,%d,%d,%d,%d,%d,%d.\n",
*(pta-7),*(pta-6),*(pta-5),*(pta-4),*(pta-3),*(pta-2),*(pta-1));
                            }
printf("All feature data received.\n");
for(t=0,pta=ta;t<nf;t++){ /* Writing out data */
   for(n=0;n<=11;n++){
  /*  printf("Data about to be transmitted to file is :- %d\n",*pta);*/
      fprintf(f2p,"%- 6d",*(pta++));
                      }
     fprintf(f2p,"\n");
                            }
printf("All feature data written out to file :- %s.\n",F2);
fclose(f2p);
goto finish;
end:
   printf("endchar received in data before all data block transferred.\n");
fclose(f2p);
finish:
printf("\n");
}
```

```
C     Program to predict and plot a bowl's area and perimeter as
C     a function of angular rotation.

      REAL AREA(200),PERI(200),ARG,MAXAR,S1,S2,WIDTH,ALPH,PERMAX
      INTEGER OPT,TXP,TYP,L,A,STEP,C1,C2,ARNDX,MARNDX,XMIN,XMAX,PENDX
      INTEGER CORN1,CORN2
      CHARACTER *70 TEXT
      CHARACTER *20 FIG
      CHARACTER *50 TXAXIS
      CHARACTER *50 TYAXIS

  502 WRITE(6,510)
  510 FORMAT('Program to Predict and Plot a Bowl s Shadow Area
     cand Perimeter as a Function of the Angle of Item Rotation.')
      WRITE(6,514)
  514 FORMAT('The main plot functions take the following forms:-')
      WRITE(6,518)
  518 FORMAT('  1/  AREA = Width * (Side1*Sine(x)+Side2*Cos(x))')
      WRITE(6,522)
  522 FORMAT('  2/  AREA = Width * ((Side1*Sine(x)+Corner1*Sine(a+x))')
      WRITE(6,524)
  524 FORMAT('                      +(Side2*Cos(x)+Corner2*Sine(a-x)))')
      WRITE(6,526)
  526 FORMAT('  3/  PERIMETER = 2*Width+2*Side1+2*Side2+f( a )')
      WRITE(6,528)
  528 FORMAT('  4/  PERIMETER = 2*Width+2*Side1+2*Side2+4*Corn1+f(a)')
      WRITE(6,530)

  530 FORMAT('Enter 1,2,3 or 4 for option to be plotted.')
      READ(6,*) OPT
      IF (OPT.NE.1.AND.OPT.NE.2.AND.OPT.NE.3.AND.OPT.NE.4) GO TO 502
      IF (OPT.EQ.2) GO TO 540
      IF (OPT.EQ.3) GO TO 560
      IF (OPT.EQ.4) GO TO 580

C     Function to take in measurements of a simple box.
      WRITE(6,532)
  532 FORMAT('Enter the variables in the following order:-')
      WRITE(6,534)
  534 FORMAT('Side1,Side2,Width (All in pixels.)')
      WRITE(6,536)
  536 FORMAT('Xmin,Xmax,Angle Step (All in degrees.)')
      READ(6,*) S1,S2,WIDTH,XMIN,XMAX,STEP
      GO TO 600

C     Function to take in relevant measurements for bowl and plate.
  540 WRITE(6,542)
  542 FORMAT('Enter the variables in the following order:-')
      WRITE(6,544)
  544 FORMAT('Side1,Side2,Width,Corner1 (All in pixels.)')
      WRITE(6,546)
  546 FORMAT('Alph,xmin,xmax,angle step (All in degrees.)')
      READ(6,*) S1,S2,WIDTH,CORN1,A,XMIN,XMAX,STEP
      ALPH=FLOAT(A)*3.142/180
      GO TO 600
  560 WRITE(6,562)
  562 FORMAT('Enter the variables in the following order:-')
      WRITE(6,564)
  564 FORMAT('Side1,Side2,Width (All in pixels)')
      WRITE(6,566)
  566 FORMAT('xmin,xmax,angle step (All in degrees.)')
      READ(6,*) S1,S2,WIDTH,XMIN,XMAX,STEP
```

Figure MF1

Figure MF1 Continued.

```
      ALPH=FLOAT(A)*3.142/180
      GO TO 600
  580 WRITE(6,582)
  582 FORMAT('Enter the variables in the following order:-')
      WRITE(6,584)
  584 FORMAT('Side1,Side2,Width,Corner1 (All in pixels)')
      WRITE(6,586)
  586 FORMAT('Alph,xmin,xmax,angle step (All in degrees.)')
      READ(6,*) S1,S2,WIDTH,CORN1,A,XMIN,XMAX,STEP
      ALPH=FLOAT(A)*3.142/180
      GO TO 600
  600 WRITE(6,610)
  610 FORMAT('Enter a title string for the plot.')
      READ(6,612) TEXT
  612 FORMAT(A)
      WRITE(6,613)
  613 FORMAT('Enter the X-axis label.')
      READ(6,614) TXAXIS
  614 FORMAT(A)
      WRITE(6,615)
  615 FORMAT('Enter the Y-axis label.')
      READ(6,616) TYAXIS
  616 FORMAT(A)
      WRITE(6,617) TEXT
  617 FORMAT('The plot title is:-',A)
C      WRITE(6,618)
C 618 FORMAT('&It is to be positioned at :-')
C      WRITE(6,620) TXP,TYP
C 620 FORMAT(2I6)
      WRITE(6,621)
  621 FORMAT('Enter a figure description.')
      READ(6,622) FIG
  622 FORMAT(A)
      WRITE(6,623)
  623 FORMAT('&The angular plot range and step is :-')
      WRITE(6,624) XMIN,XMAX,STEP
  624 FORMAT(3I6)
      IF (OPT.EQ.3) GO TO 650
      IF (OPT.EQ.4) GO TO 680

C      Generate the area values.
      ARNDX=0
      MAXAR=0
      DO 628 L=XMIN,XMAX,STEP
        ARNDX=ARNDX+1
        ARG=FLOAT(L)*3.142/180
      IF(OPT.EQ.1)AREA(ARNDX)=WIDTH*(ABS(S1*SIN(ARG))+ABS(S2*COS(ARG)))
      IF(OPT.EQ.2)AREA(ARNDX)=WIDTH*(ABS(S1*SIN(ARG))+ABS(S2*COS(ARG))+
     c ABS(CORN1*SIN(ALPH+ARG))+ABS(CORN1*SIN(ALPH-ARG)))
        IF(AREA(ARNDX).GT.MAXAR) MAXAR=AREA(ARNDX)
        WRITE(6,626) L,ARNDX,AREA(ARNDX)
  626   FORMAT(I8,I8,F10.2)
  628 CONTINUE
      MARNDX=ARNDX
      ARNDX=1
      GO TO 700

C      First perimeter plot.
  650 CONTINUE
      PENDX=0
```

Figure MF1 Continued.

```
      PERMAX=0
      DO 670 L=XMIN,XMAX,STEP
        PENDX=PENDX+1
        IF(L.LE.3.OR.L.GE.177)PERI(PENDX)=2*WIDTH+2*S1
        IF(L.GT.3.AND.L.LT.87)PERI(PENDX)=2*WIDTH+2*S1+2*S2
        IF(L.GE.87.AND.L.LE.93)PERI(PENDX)=2*WIDTH+2*S2
        IF(L.GT.93.AND.L.LT.177)PERI(PENDX)=2*WIDTH+2*S1+2*S2
        IF(PERI(PENDX).GT.PERMAX) PERMAX=PERI(PENDX)
        WRITE(6,652) L,PENDX,PERI(PENDX)
  652   FORMAT(I8,I8,F10.2)
  670 CONTINUE
  672 GO TO 690

C      Second perimeter plot.
  680 CONTINUE
      PENDX=0
      PERMAX=0
      DO 690 L=XMIN,XMAX,STEP
        PENDX=PENDX+1
        IF(L.LE.3.OR.L.GE.177) THEN
          PERI(PENDX)=2*WIDTH+2*S2+4*CORN1
        ELSE IF(L.LT.87.AND.((L+A).LT.87.OR.(L+A).GT.93)) THEN
          PERI(PENDX)=2*WIDTH+2*S1+2*S2+4*CORN1
        ELSE IF(L.LT.87.AND.((L+A).GE.87.AND.((L+A).LE.93))) THEN
          PERI(PENDX)=2*WIDTH+2*S1+2*S2+2*CORN1
        ELSE IF(L.GE.87.AND.L.LE.93) THEN
          PERI(PENDX)=2*WIDTH+2*S1+4*CORN1
        ELSE IF(L.GT.93.AND.((L+A).GE.177.AND.((L+A).LE.183))) THEN
          PERI(PENDX)=2*WIDTH+2*S1+2*S2+2*CORN1
        ELSE IF(L.GT.93.AND.((L+A).LT.177.OR.((L+A).GT.183))) THEN
          PERI(PENDX)=2*WIDTH+2*S1+2*S2+4*CORN1
        END IF
        IF(PERI(PENDX).GT.PERMAX) PERMAX=PERI(PENDX)
        WRITE(6,682) L,PENDX,PERI(PENDX)
  682   FORMAT(I8,I8,F10.2)

C      Prepare the perimeter data for the plot routine.
  690 CONTINUE
      DO 700 L=1,200
         AREA(L)=PERI(L)
         WRITE(6,692) L,AREA(L)
  692    FORMAT(I8,F10.2)
  700 CONTINUE
      IF(OPT.EQ.1.OR.CPT.EQ.2) MAXAR = 1300
      IF(OPT.EQ.3.OR.CPT.EQ.4) MAXAR = 300
      ARNDX=1
      GO TO 800

C      Points are now ready for plotting.
  800 CONTINUE
      CALL PAPER(1)
      CALL PSPACE(0.1,0.8,0.1,0.8)
      CALL MAP(FLOAT(XMIN),FLOAT(XMAX),0,MAXAR)
      CALL BORDER
      CALL AXES
      CALL CTRMAG(20)
      CALL POSITN(FLOAT(XMIN),AREA(1))
      DO 850 L=XMIN+STEP,XMAX,STEP
         ARNDX=ARNDX+1
         CALL JOIN(FLOAT(L),AREA(ARNDX))
```

```
 850 CONTINUE

C      LABEL THE PLOT
C      CALL PLOTCS(FLOAT(TXP),FLOAT(TYP),TEXT)
C      CALL PLOTCS(FLOAT(XMAX),FLOAT(TYP),FIG)
      CALL PSPACE(0.0,1.0,0.0,1.0)
      CALL MAP(0.0,1.0,0.0,1.0)
      CALL PCSCEN(0.65,0.85,TEXT)
      CALL PLOTCS(0.9,0.75,FIG)
      CALL CTRMAG(15)
      CALL PLOTCS(0.35,0.03,TXAXIS)
      CALL CTRORI(90.0)
      CALL PLOTCS(0.03,0.45,TYAXIS)
      CALL GREND
      STOP
      END
```

Figure MF1 Continued.

Figure MF2

```
C     Program to read in item data and crossplot any two features
C     of my choice from a feature-data file.
C
      INTEGER DATA(300,12)
      INTEGER OPT,ND,IN,IL,L,INX,INY,TXP,TYP,X1,X2,Y1,Y2
      REAL SX1,SX2,SY1,SY2,XOFFS,YOFFS
      CHARACTER *70 TEXT
      CHARACTER *20 FIG
      CHARACTER *50 TXAXIS
      CHARACTER *50 TYAXIS
C
    3 WRITE(6,10)
   10 FORMAT('Select the item data file to be read in.')
      WRITE(6,12)
   12 FORMAT('Enter a 1 for a bowl file.')
      WRITE(6,14)
   14 FORMAT('Enter a 2 for a cup file.')
      WRITE(6,15)
   15 FORMAT('Enter a 3 for a plate file.')
C
C     READ IN THE CHOSEN SOURCE DATA FILE.
      READ(6,16) OPT
   16 FORMAT(I1)
      IF(OPT.NE.1.AND.OPT.NE.2.AND.OPT.NE.3) GO TO 3
      WRITE(6,18)
   18 FORMAT('Enter the number of cases to be read in.')
      READ(6,20) ND
   20 FORMAT(I3)
      IF(OPT.EQ.2) GO TO 50
      IF(OPT.EQ.3) GO TO 82
      WRITE(6,22)
   22 FORMAT('About to read in the bowl file.')
      WRITE(6,24) ND
   24 FORMAT('Number of cases is :- ',I3)
      OPEN(7,FILE='SHIPNB1',STATUS='OLD')
      DO 40 L=1,ND
            READ(7,*) (DATA(L,IL),IL=1,12)
   40 CONTINUE
   42 GO TO 100

C     EQUIVALENT STATEMENTS FOR THE  CUP.
   50 WRITE(6,52)
   52 FORMAT('About to read in the cup file.')
      OPEN(7,FILE='SHIPNC1',STATUS='OLD')
      DO 60 L=1,ND
            READ(7,*) (DATA(L,IL),IL=1,12)
   60 CONTINUE
      GO TO 100

C     EQUIVALENT STATEMENTS FOR PLATE.
   82 WRITE(6,84)
   84 FORMAT('About to read in the plate file.')
      OPEN(7,FILE='NE1',STATUS='OLD')
      DO 90 L=1,ND
            READ(7,*) (DATA(L,IL),IL=1,12)
   90 CONTINUE
  100 CONTINUE
  120 CONTINUE

C     ALLOW USER TO SPECIFY AXES.
      WRITE(6,126)
  126 FORMAT('&Enter two numbers abscissa first then ordinate')
```

Figure MF2 Continued.

```
      WRITE(6,128)
  128 FORMAT(' from the following list.')
      WRITE(6,130)
  130 FORMAT('Enter 1 for Index.')
      WRITE(6,132)
  132 FORMAT('Enter 2 for Perimeter.')
      WRITE(6,134)
  134 FORMAT('Enter 3 for Area.')
      WRITE(6,136)
  136 FORMAT('Enter 4 for Compaction.')
      WRITE(6,138)
  138 FORMAT('Enter 5 for Total.')
      WRITE(6,140)3
  140 FORMAT('Enter 6 for Minmax.')
      WRITE(6,142)
  142 FORMAT('Enter 7 for Endmid.')
      WRITE(6,144)
  144 FORMAT('Enter 8 for MinMid.')
      WRITE(6,146)
  146 FORMAT('Enter 9 for MidMax.')
      WRITE(6,148)
  148 FORMAT('Enter 10 for DtoMin.')
      WRITE(6,149)
  149 FORMAT('Enter 11 for DtoMid.')
      WRITE(6,150)
  150 FORMAT('Enter 12 for DtoMax.')
      READ(6,*) INX,INY
      WRITE(6,154)
  154 FORMAT('Enter a title string for the plot.')
      READ(6,156) TEXT
  156 FORMAT(A)
      WRITE(6,157)
  157 FORMAT('Enter the X-axis label.')
      READ(6,158) TXAXIS
  158 FORMAT(A)
      WRITE(6,159)
  159 FORMAT('Enter the Y-axis label.')
      READ(6,160) TYAXIS
  160 FORMAT(A)
      WRITE(6,161)
  161 FORMAT('Enter a figure decription.')
      READ(6,162) FIG
  162 FORMAT(A)
      WRITE(6,163)
  163 FORMAT('Enter the plot ranges of x then y coordinates.')
      READ(6,*) X1,X2,Y1,Y2
      WRITE(6,164)
  164 FORMAT('&The plot title is:-')
      WRITE(6,166) TEXT
  166 FORMAT(A)
      WRITE(6,172)
  172 FORMAT('&The x plot range is:-')
      WRITE(6,174) X1,X2
  174 FORMAT(2I6)
      WRITE(6,176)
  176 FORMAT('&The y plot range is:-')
      WRITE(6,178) Y1,Y2
  178 FORMAT(2I6)

C     PRE-SCALING OF PLOT AREA.
```

Figure MF2 Continued.

```
      SX1=((X2-X1)*0.5)

C      PLOTTING SECTION OF THE PROGRAM.
      CALL PAPER(1)
      CALL CTRMAG(20)
      CALL PSPACE(0.1,0.8,0.1,0.8)
      CALL MAP(FLOAT(X1),FLOAT(X2),FLOAT(Y1),FLOAT(Y2))
      CALL BORDER
      CALL AXES
      CALL POSITN(FLOAT(DATA(1,INX)),FLOAT(DATA(1,INY)))
      CALL PLOTNC(FLOAT(DATA(1,INX)),FLOAT(DATA(1,INY)),162)
      DO 200 L=2,ND
         CALL JOIN(FLOAT(DATA(L,INX)),FLOAT(DATA(L,INY)))
         CALL PLOTNC(FLOAT(DATA(L,INX)),FLOAT(DATA(L,INY)),162)
  200 CONTINUE

C       LABEL THE PLOT
C       CALL PCSCEN(SX1,FLOAT(Y2),TEXT)
C       CALL PLOTCS(FLOAT(X2),FLOAT(Y2),FIG)
C
       CALL PSPACE(0.0,1.0,0.0,1.0)
       CALL MAP(0.0,1.0,0.0,1.0)
C
       CALL PCSCEN(0.65,0.85,TEXT)
       CALL PLOTCS(0.9,0.75,FIG)
       CALL CTRMAG(15)
       CALL PLOTCS(0.35,0.03,TXAXIS)
       CALL CTRORI(90.0)
       CALL PLOTCS(0.03,0.45,TYAXIS)
C
       CALL GREND
       CLOSE (7)
      STOP
      END
```